

4.3

*i2 Analyze
Documentation*

IBM

Note

Before using this information and the product it supports, read the information in [“Notices” on page 479.](#)

This edition applies to version 4, release 3, modification 4 of i2 Analyze (product number 5725-G22) and to all subsequent releases and modifications until otherwise indicated in new editions. Ensure that you are reading the appropriate document for the version of the product that you are using. To find a specific version of this document, navigate to the [i2 Analyze](#) documentation and select the appropriate version.

© **Copyright International Business Machines Corporation 2018, 2021.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices and contact information.....	1
Copyright.....	1
Notices.....	1
Contacting IBM Support.....	3
Understanding i2 Analyze.....	5
Components.....	6
Chart Store.....	6
i2 Connect gateway.....	7
Information Store.....	7
Data model.....	8
The i2 Analyze data model.....	8
Data in i2 Analyze records.....	14
Storing charts in i2 Analyze.....	19
Security model.....	20
Security dimensions.....	20
Security permissions.....	21
Security model example.....	23
Security architecture.....	24
Supplied security implementation.....	25
Logging and auditing.....	26
Deploying i2 Analyze.....	29
Deployment types.....	29
Creating an example deployment.....	30
Creating an example with the Chart Store.....	30
Creating an example with the Information Store.....	31
Creating an example with the i2 Connect gateway.....	33
Creating an example with the Chart Store and the i2 Connect gateway.....	35
Creating an example with the Information Store and the i2 Connect gateway.....	37
Creating a production deployment.....	39
Deployment phases and environments.....	40
Schema development environment.....	41
Configuration development environment.....	45
Pre-production environment.....	48
Test environment.....	50
Production environment.....	52
Deployment resources.....	54
Deployment topologies.....	54
The i2 Analyze deployment toolkit.....	57
i2 Analyze deployment settings.....	67
i2 Analyze with high availability.....	74
i2 Analyze deployment commands.....	85
Troubleshooting the deployment process.....	88
Connecting clients.....	92
Removing a deployment of i2 Analyze.....	92
Best practices.....	93
Installing i2 Analyze.....	94

Software prerequisites.....	94
Configuring i2 Analyze.....	101
Configuring the i2 Analyze application.....	101
i2 Analyze schemas.....	102
Security schema.....	108
Item type security.....	119
User security.....	126
Controlling access to features.....	142
System limits and variables.....	148
Search.....	159
Matching.....	201
Geospatial mapping.....	213
Source references.....	224
Securing i2 Analyze.....	229
Adding more data sources.....	262
Back up and restore.....	263
Administering high availability.....	272
Configuration resources.....	282
Connecting to external data sources.....	306
i2 Analyze and the i2 Connect gateway.....	306
System architecture with the i2 Connect gateway.....	308
System security with the i2 Connect gateway.....	309
Creating a connector to an external data source.....	310
Extending a system that queries external sources.....	320
Valid formats for i2 Connect data values.....	334
IBM i2 Analyze Schema Designer.....	337
Notices and contact information.....	338
Schemas.....	341
Schema validation.....	343
Creating schemas.....	344
Semantic types and charting schemes.....	348
Changing the grading scheme.....	356
Changing link strengths.....	357
Glossary.....	357
Managing data in i2 Analyze.....	367
Ingesting data into the Information Store.....	367
Information Store data ingestion.....	367
Information Store data correlation.....	416
Information Store data deletion.....	437
About this guide.....	437
Contacting IBM Support.....	438
Overview.....	438
Deleting shared records.....	439
Deleting records by rule.....	441
Retrieving chart metadata.....	455
Determining when a chart was last accessed.....	455
Determining which users uploaded a chart.....	456
Sending alerts to i2 Analyze users.....	456
Creating alerts.....	456
Deleting alerts.....	462
Scheduling custom server tasks.....	462

Creating a custom task.....	462
Deploying a custom task.....	463
Monitoring a custom task.....	464
Upgrading i2 Analyze.....	465
Upgrade paths.....	465
Upgrading to i2 Analyze 4.3.5.....	465
Upgrading the ETL toolkit.....	467
Upgrading IBM i2 Analyst's Notebook Premium.....	467
Upgrade resources.....	467
Upgrading an i2 Analyze deployment.....	467
Configuration and database changes.....	470
Information Store processing after you upgrade i2 Analyze.....	474
Upgrading a customized Information Store.....	475
Download pdfs.....	477
Notices.....	479
Trademarks.....	480

Notices and contact information

Copyright

© Copyright IBM Corporation 2012, 2021.

All rights reserved. May only be used pursuant to an IBM software license agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of IBM Corporation. IBM Corporation grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the IBM Corporation copyright notice. No other rights under copyright are granted without prior written permission of IBM Corporation. The document is not intended for production and is furnished "as is" without warranty of any kind.

All warranties on this document are hereby disclaimed, including the warranty of non-infringement and the implied warranties of merchantability and fitness for a particular purpose.

U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited

Hursley House

Hursley Park

Winchester, Hants, SO21 2JN

UK

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

ArcGIS, EDN, StreetMap, @esri.com, and www.esri.com are trademarks, registered trademarks, or service marks of Esri in the United States, the European Community, or certain other jurisdictions.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.

Data Storage

IBM provides the ability to store data in IBM i2 products for investigative purposes. The security and handling of data that is added into IBM i2 products remains the responsibility of the customer. Be aware that data that is stored in IBM i2 Analyst's Notebook chart formats (such as *.anb and *.anbx) is not encrypted, therefore additional measures may be needed to ensure data security when storing and sharing files of these types.

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users to resolve problems with the product.

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM software subscription and support contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

To contact IBM Support about a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM Support topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
 - Online through the IBM Support Portal at <https://www.ibm.com/mysupport/>. You can open, update, and view all of your service requests from the Service Request portlet on the Service Request page.
 - By phone. For the phone number to call in your region, see the Directory of worldwide contacts web page at <https://www.ibm.com/planetwide/>

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

Understanding i2 Analyze

i2[®] Analyze is an integrated set of secure services and stores that provide search, analysis, and storage of intelligence data to authorized users. It presents these functions through the web and desktop applications that connect to it.

When deployed with all components active, i2 Analyze provides key features to its users:

- A structured store for intelligence data with services that enable bulk and targeted create, retrieve, update, and delete operations.
- An extensible framework for connecting to and retrieving intelligence data from sources external to the platform.
- A searchable store for Analyst's Notebook charts that analysts can use for remote storage, and for sharing work with their peers.
- A pervasive data model that is optimized for exploring and visualizing the relationships between records.
- A security model and architecture that together ensure the security of data in motion and at rest.

i2 Analyze enables these features through a range of technologies:

- The i2 Analyze services are deployed in an application on a WebSphere Liberty application server.
- Searching and returning results from intelligence data uses the Apache Solr search index platform.
- The stores for intelligence data and Analyst's Notebook charts are created in a database management system.

The precise feature set of a particular deployment of i2 Analyze depends on the components that you decide to include, which depends in turn on the IBM i2 software that you use it with. The different components of i2 Analyze are subject to licensing agreements.

Understanding

i2 Analyze: Data model

The i2 Analyze data model governs the structure of the data that i2 Analyze processes. That structure, consisting of entities and links and the properties they contain, is fundamental to the analytical and presentational functionality provided by other offerings that i2 Analyze is part of, for example IBM i2 Analyst's Notebook Premium and IBM i2 Enterprise Insight Analysis.

i2 Analyze: Security model

All data in i2 Analyze can be secured so that only the users who are supposed to interact with it are able to do so. Using the i2 Analyze security model, you can decide what access users have to records and features, based on their membership of user groups.

i2 Analyze: Logging and auditing

i2 Analyze provides mechanisms for logging two types of information that the system generates during normal execution. You can control what information is sent to the system logs, and audit the commands that users invoke.

Configuration tasks

Configuring i2 Analyze

During implementation of a production deployment, you need to modify the original base deployment to match the needs of your organization. When the i2 Analyze deployment is in use, you can make further configuration changes to adjust to changing needs, and to administer the system.

Troubleshooting and support

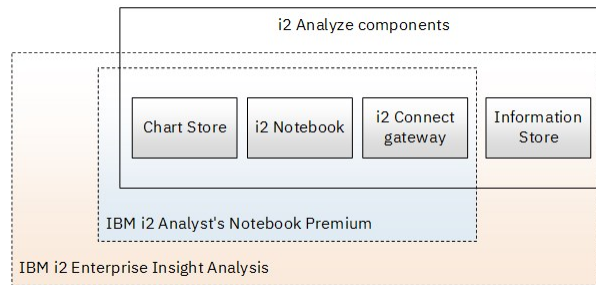
i2 Analyze support page

IBM Software Support home page

Components

The data and security models that i2 Analyze provides are a constant in all deployments. Its other features are due to components that you can choose to deploy according to your requirements and the terms of your license.

IBM provides i2 Analyze as a part of other offerings. For example, IBM i2 Analyst's Notebook Premium and IBM i2 Enterprise Insight Analysis. These offerings entitle you to deploy different components of i2 Analyze.



- For any of the offerings, you can deploy i2 Analyze with the Chart Store, so that users can store and share their charts securely.
- You can also deploy i2 Analyze with the i2 Connect gateway, which allows you to develop and deploy connectors to data sources outside your organization.
- For Enterprise Insight Analysis only, you can also deploy the Information Store, in which users can retain, process, and analyze intelligence data.

Chart Store

The Chart Store has two purposes in a deployment of i2 Analyze. First, it provides Analyst's Notebook Premium users with server-based, secure storage for their charts. Second, in a deployment that does not also include the Information Store, it supports the operation of the i2 Notebook web client.

Users of Analyst's Notebook Premium can upload charts to a Chart Store as easily as they can save them to their workstations. When they do so, they can choose to share charts with their colleagues, or take advantage of i2 Analyze's features for organizing, indexing, and securing charts:

- i2 Analyze authenticates users at login and then uses authorization to control access to all stored charts
- Charts in the Chart Store are subject to the i2 Analyze security model, enabling per-team or per-user access
- i2 Analyze indexes both the contents and the metadata of uploaded Analyst's Notebook charts, so that users can search and filter large numbers of charts quickly and easily

Note: For each chart in the store, the generated index contains information from i2 Analyze records and Analyst's Notebook chart items. i2 Analyze does not index the contents of any documents that are embedded in a chart.

- The chart metadata itself is configurable, so that the options for categorizing and searching charts reflect the users' domain.

Users of the i2 Notebook web client can search for and visualize data from an Information Store and any external sources that are connected through the i2 Connect gateway. However, not all deployments of i2 Analyze include both of those components. In a deployment that does not have

an Information Store, the i2 Notebook web client requires the Chart Store, which provides short-term storage for the web charts that users create.

For information on deploying i2 Analyze with the Chart Store, see the [deployment documentation](#).

i2 Connect gateway

When you deploy i2 Analyze with the i2 Connect gateway, you gain the option to develop or purchase connectors that can acquire intelligence data from external sources. When a user of Analyst's Notebook Premium or the i2 Notebook web client makes a request to an external data source, the connector converts results from their original format into the entities, links, and properties of i2 Analyze records.

When i2 Analyze accesses data through a connector, the source is not modified. After the data is vetted and verified, and provided that the new or modified entities and links have compatible types, Analyst's Notebook Premium users can upload the records to an Information Store.

For each external data source, an organization can choose between setting up a connector to access the data in the external location, and ingesting the data into an Information Store. The choice depends on a number of factors, including the following considerations:

Availability

A connector is typically useful where a data source is constantly available. If a data source is not always available, ingestion to the Information Store provides users with constant access to a snapshot of the data.

Currency

If an external data source is updated very frequently, and it is desirable to have the most current data returned at each request, a connector can be used to achieve this. If regular refreshes are appropriate for the data, but moment-to-moment currency is not required, the data source could instead be ingested to the Information Store on a regular schedule.

Quantity

The Information Store is designed for high-volume data. However, an organization might not want to create a duplicate copy of a very large external data source that can be made available using a connector.

Terms of use

For reasons of security, privacy, or commercial interest, a third-party provider of data might not permit the organization to store a copy of the data source in their own environment. In this situation, a connector must be used to access the external data source.

For more information about how the i2 Connect gateway and connectors provide access to external data, see [Connecting to external data sources](#). For examples of creating and deploying connectors, see the open-source project at <https://github.com/IBM-i2/Analyze-Connect>.

Information Store

The Information Store is a secure, structured store for intelligence data that analysts and other users can search and visualize. i2 Analyze provides services that enable bulk and targeted create, retrieve, update, and delete operations on Information Store data. In deployments that include it, the Information Store also fulfills the functions of the “Chart Store” on page 6.

Records in the Information Store can be loaded automatically from data sources external to i2 Analyze, or they can be imported or created by individual Analyst's Notebook Premium users. To cope with the different requirements of bulk and targeted operations, the store distinguishes between system- and analyst-governed data:

- System-governed data is loaded into the Information Store in bulk. i2 Analyze includes a toolkit that accelerates the process of extracting and transforming data from external sources and loading it into the Information Store. You can arrange for correlation to take place during this process, and for subsequent loads from the same source to update or delete data that the Information Store has seen before.
- Analyst-governed data is added to the Information Store through Analyst's Notebook Premium, when users upload records that they've imported or created directly on their charts. To optimize that process, i2 Analyze provides matching functionality, so that users are alerted when records that they create match records that are already in the Information Store.

Subject to authorization, Analyst's Notebook Premium users can modify analyst-governed records in the Information Store, but system-governed records are read-only. Apart from that distinction, all records in the Information Store can be searched for, compared, and analyzed in exactly the same way, regardless of governance.

Users of the i2 Investigate and i2 Notebook web clients cannot modify any records in the Information Store, but they can search for and visualize records from the store using the tools that those applications provide.

To understand the structure of data in the Information Store, see [The i2 Analyze data model](#). To learn how to deploy i2 Analyze with the Information Store, see [Creating an example with an Information Store](#) and [Creating a production deployment](#). For more information about using the ETL toolkit to populate the Information Store with system-governed data, see [Ingesting data into the Information Store](#).

Data model

The i2 Analyze data model governs the structure of the data that i2 Analyze processes. That structure, consisting of entities and links and the properties they contain, is fundamental to the analytical and presentational functionality provided by other offerings that i2 Analyze is part of, for example IBM i2 Analyst's Notebook Premium and IBM i2 Enterprise Insight Analysis.

The i2 Analyze data model

The i2 Analyze data model is the foundation on which the abilities of the platform to manage, process, and visualize data are based. On the server and in the user interface, IBM i2 software models data in terms of entities, links, and properties (ELP).

- An *entity* represents a real-world object, such as a person or a car.
- A *link* associates two entities with each other. For example, a Person entity might be associated with a Car entity through an Owns link.
- A *property* stores a value that characterizes an entity or a link. For example, a Person entity might have properties that store their given name, their surname, their date of birth, and their hair color.

Every deployment of i2 Analyze contains descriptions of the kinds of entities and links that can appear in the data for that deployment. These descriptions also state exactly what properties the entities and links can have, and define the relationships that can exist between entities and links of different kinds.

In i2 Analyze, [schemas](#) provide these deployment-specific descriptions of the data model:

- If your deployment includes an Information Store, a schema determines both its structure and the shape that data must have in order for the Information Store to process it.
- If your deployment includes the i2 Connect gateway, each connector can have a schema that describes the shape of the data that it provides. Additionally, the gateway can have one or more schemas that describe the data for several related connectors.

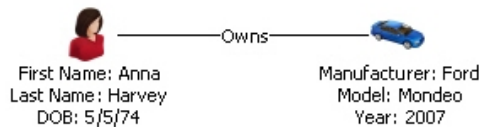
To create a schema for an i2 Analyze deployment, you must examine the data that is likely to be available for analysis, and understand how that data is used during an investigation. Understanding the aims of your investigations is key to helping you to organize data effectively.

ELP relationships in i2 Analyze

Depending on the nature of the data that you want to process in your deployment of i2 Analyze, you might need to shape it to the ELP (entity, link, property) format. Putting data into ELP format enables many of the analytical functions that i2 Analyze provides.

The simplest ELP relationship involves two entities that are connected with a single link. These kinds of relationships are the building blocks for networks that contain groups and chains of entities with any number of links between them.

In i2 Analyze, a simple relationship that involves two entities, a link, and their properties can be visualized like this example:



Note: Because of the way that these relationships appear in visualizations, the structure is sometimes called a *dumbbell*.

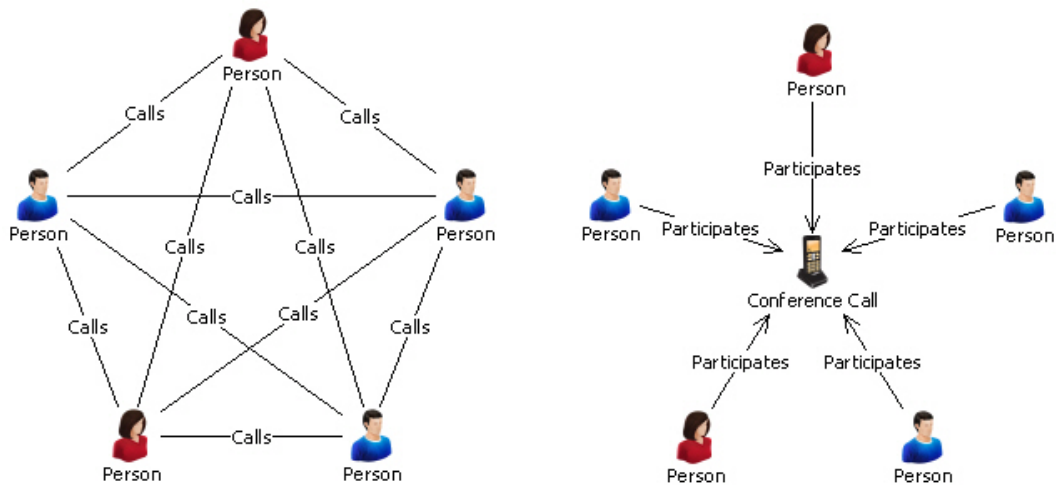
Some of the information that users see in a relationship like this one comes from the data itself:

- For the entity on the left, the data includes the property values "Anna", "Harvey", and "5/5/74".
- Similarly, for the entity on the right, the property values include "Ford", "Mondeo", and "2007".
- The data for the link includes a way of identifying the two entities that it connects.

The remainder of the information in the example comes from definitions in an i2 Analyze schema:

- The default icons for the entities, and the names ("First Name", "Manufacturer") and logical types of their properties, are all defined in an i2 Analyze schema.
- The default label for the link ("Owns") is also defined in an i2 Analyze schema.

In practice, it can be best to make links lightweight and use intermediate entities to model the details of complex associations. Among other things, this approach allows improved modeling of multi-way associations, such as a conference call that has multiple participants. The following diagram shows the difference:



To align your data with an i2 Analyze schema, you must resolve it into the component parts of ELP relationships. If your source is a relational database, it is possible that your tables each correspond to particular kinds of entities, while foreign key relationships form the basis for particular kinds of links. If your source is a delimited text file, it is possible that rows in the file contain the data for one or more entities and links.

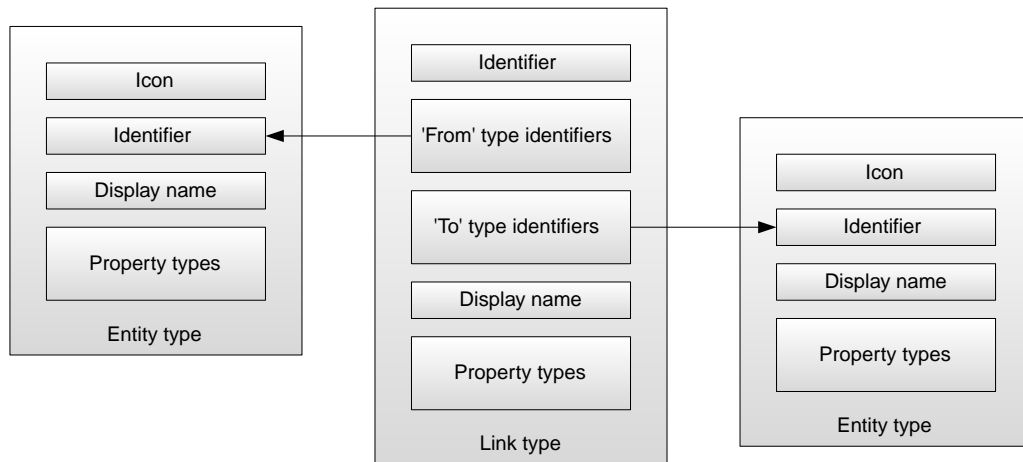
Entity types, link types, and property types

In the i2 Analyze data model, *entity types* and *link types* that determine what entities and links can appear in the data to which the schema applies. The *property types* of those entity and link types determine what properties the entities and links can have.

Entity types and link types

In an i2 Analyze schema, entity types and link types have similar definitions. Among several common features, entity types and link types both have:

- Identifiers, so that types can refer to each other in rules about their relationships
- Names that users see when they interact with entities and links of particular types
- Definitions of the properties that entities and links of particular types can contain

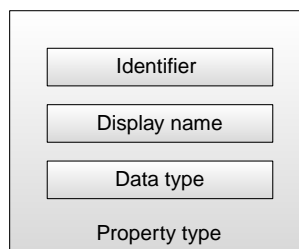


As well as the common features, each *entity* type contains the icon that represents entities with that type in visualizations. *Link* types do not contain icons, but they do contain lists of 'from' and 'to' entity type identifiers. For a link that has a particular link type, these lists determine what entity types the entities at each of the link can have. For example, a link of type Calls might be valid in both directions between two entities of type Person. An Owns link might be valid between a Person and a Car, but would not be allowed in the opposite direction.

For an i2 Analyze schema to be valid, any identifiers that appear in the 'from' and 'to' lists of link types must also appear as the identifiers of entity types.

Property types

In an i2 Analyze schema, entity types and link types both contain property types. For an entity or link record that has a particular type, the property types specify the names and the logical types of the properties that the entity or link can have.



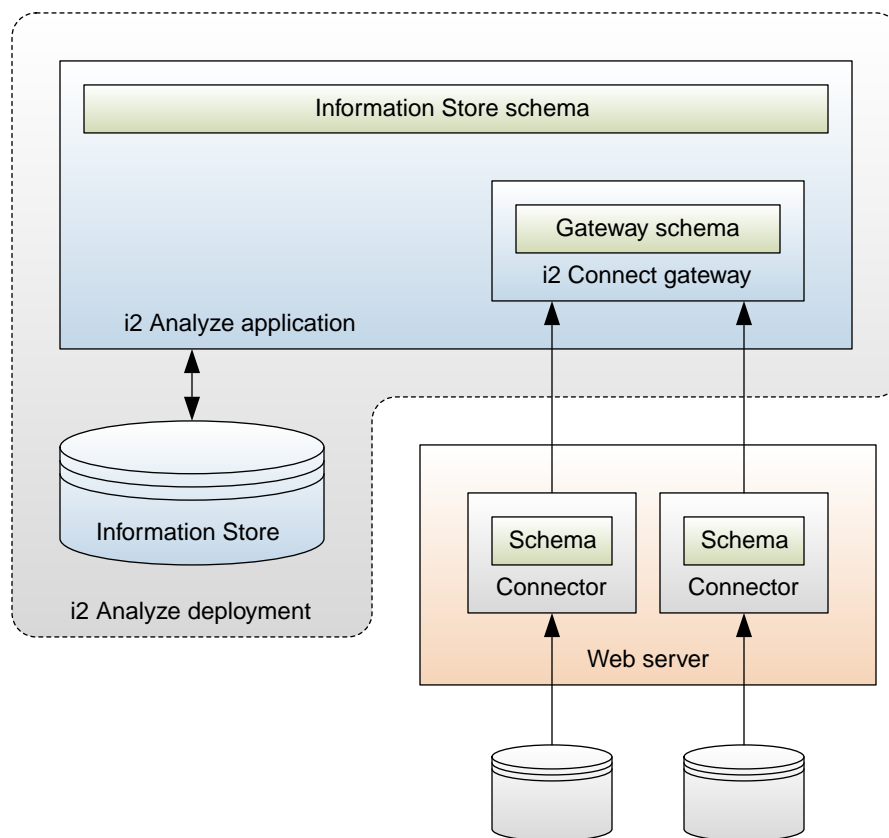
Note: This representation is simplified. A property type can also specify a list of possible values for properties of that type. Furthermore, it can declare whether a property of that type is mandatory for an entity or a link that has the containing type.

i2 Analyze schemas

An i2 Analyze schema is a statement about the categories of data that a deployment can work with, and about the relationships that can exist between data in different categories. As a result, i2 Analyze schemas are responsible for what data looks like when it is visualized; for the types of analysis that can users can perform; and for the structures in which data is stored for retrieval.

Kinds of schema

All deployments of i2 Analyze must contain at least one schema. Some deployments can contain several schemas. All schemas perform the same role, but which kinds of schema you need depends partly on how you want your deployment to behave, and partly on which components of i2 Analyze it uses.



Information Store schema

If your i2 Analyze deployment includes an Information Store, then it requires an Information Store schema. If it includes *only* an Information Store, then the Information Store schema is the only one you need.

i2 Analyze uses the Information Store schema to create the Information Store database. When you later load data into the Information Store, you must shape the data so that it matches the entity and link types in the schema. And if a client application wants to import data with the aim of uploading it to the Information Store, then it uses the schema to ensure compatibility.

Important: Because of its relationship to the structure of the database, an Information Store schema is the most difficult to modify in a production deployment. Only additive changes to the schema are permitted.

Gateway schema

Gateway schemas are relevant only to deployments of i2 Analyze that include the i2 Connect gateway. When the gateway is present, gateway schemas are optional.

The data that connectors return from external sources must have the same ELP structure as other data in i2 Analyze. Sometimes, you can make data from a connector use the types that an Information Store schema defines. Sometimes, a connector returns data with unique types, and defines a schema of its own. But sometimes, you might have several connectors whose data shares a set of common types. In this situation, you can create a gateway schema and arrange for all the connectors to use the types that the gateway schema defines.

Note: Because gateway schemas are not tied to the structure of data storage, they are relatively easy to modify and redeploy. Developing and testing a gateway schema can be a convenient way to create a schema that you plan eventually to use for the Information Store.

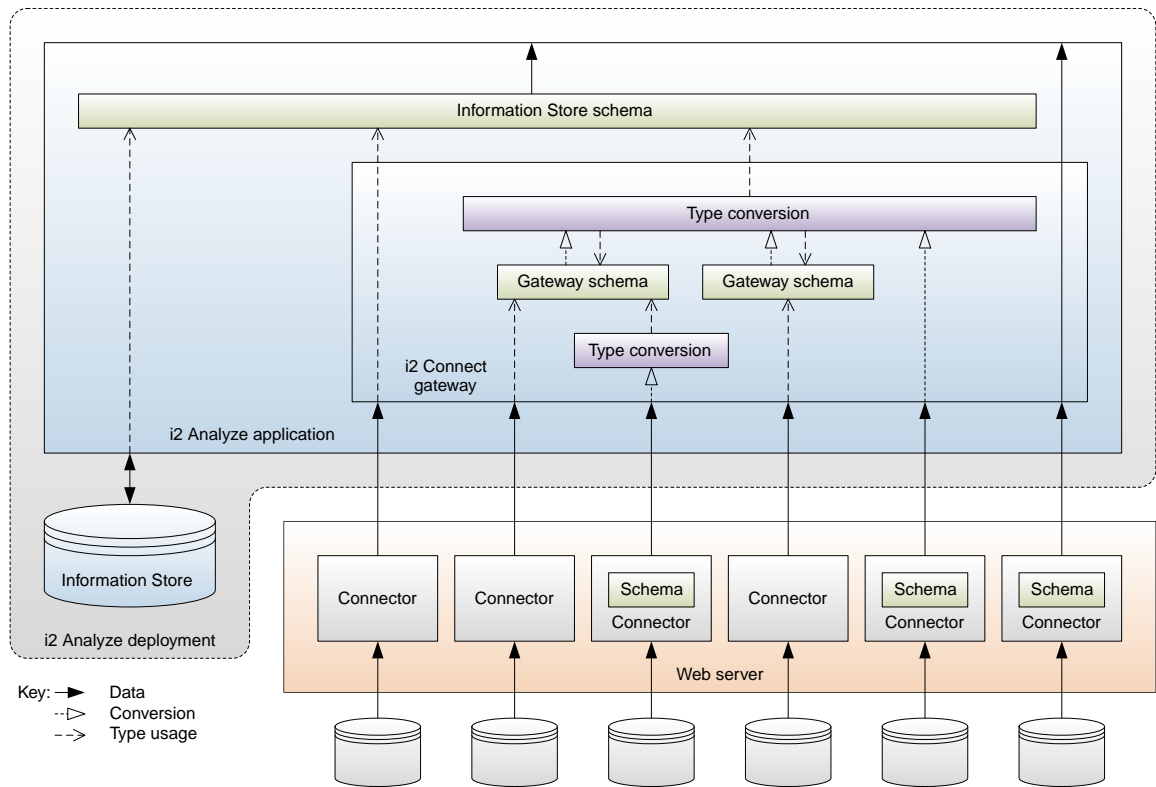
Connector schema

All data in an i2 Analyze deployment must use entity and link types from a schema. If a connector does not or cannot use types from an Information Store schema or a gateway schema, then it must provide its own definitions of the types that it uses.

A connector schema can be appropriate when you are prototyping a connector to a new data source, or when you know that data from a particular source is subject to frequent changes. Alternatively, you might be using or creating a connector that is designed for use in multiple i2 Analyze deployments. In that case, it can be helpful for the connector to come with definitions of its own types.

Schema type conversion

An i2 Analyze deployment that includes the i2 Connect gateway and connectors to external data sources is likely to involve several schemas. Client software can visualize data that uses types from any schema, display its property values, and subject it to structural analysis. However, data can be uploaded to an Information Store only if it uses types from that store's schema. And users can perform comparative analysis only between data that uses the same types.



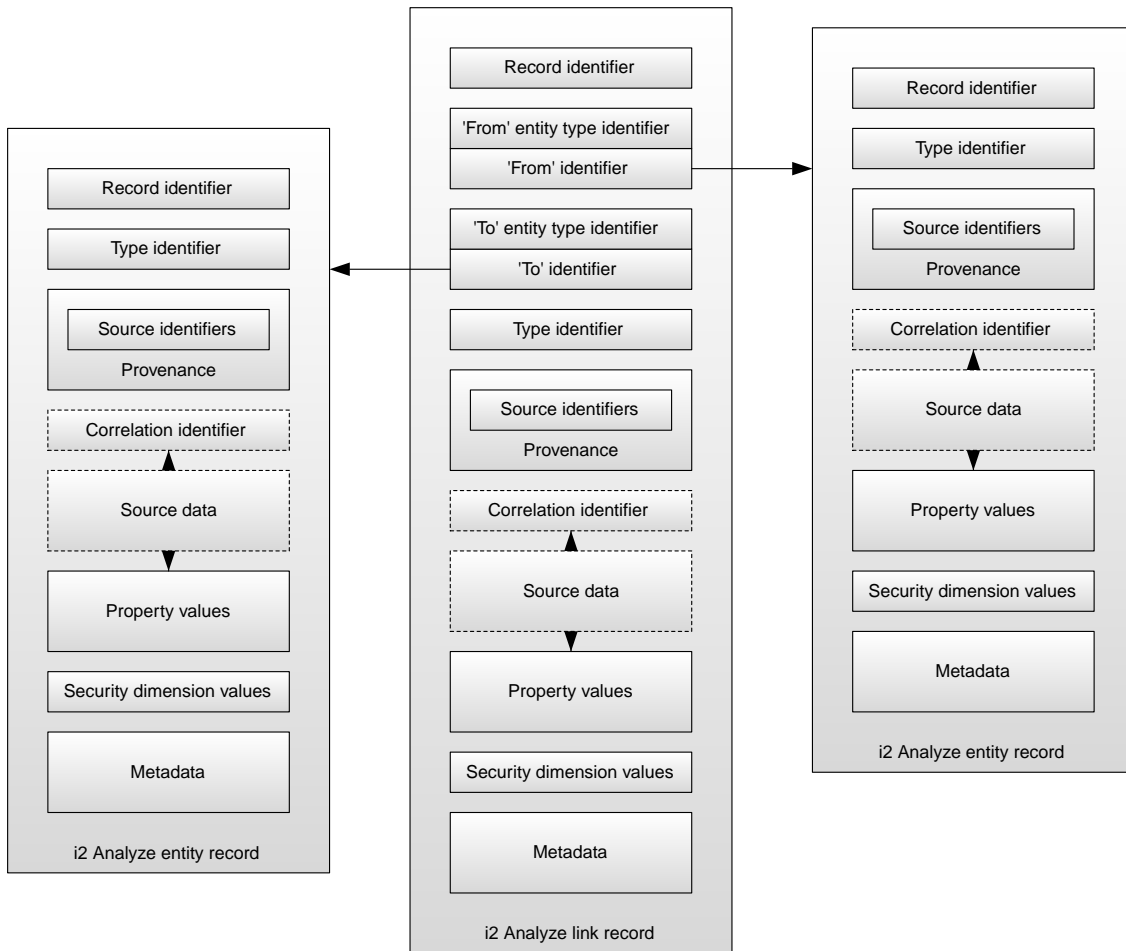
To enable i2 Analyze to treat data from all sources similarly, you can arrange for records to be converted from one type to another as they move through the system. To enable this type conversion, you provide one-to-one mappings between types from different schemas. For example, if you map the types from a connector schema (or a gateway schema) to types in the Information Store schema, then you can upload data from that connector to the Information Store. If you map types from several connector schemas to the types in a gateway schema (or the types in one gateway schema with those in another), it becomes possible to use data from one connector as seeds for searches of another.

Data in i2 Analyze records

i2 Analyze deployments use *i2 Analyze records* to realize the entity and link types that schemas define. i2 Analyze records contain the property data for entities and links, plus metadata that enhances the analysis that users can carry out.

An i2 Analyze schema defines the types that determine what i2 Analyze records can represent. Every i2 Analyze record has a type that an i2 Analyze schema defines. If a record has a link type, then the record represents a link – it is a *link record*. If a record has an entity type, then it is an *entity record*.

This diagram shows how entity and link records compare, and how they are related to each other. It also introduces some other features of the data in i2 Analyze records.



Note: The diagram contains some simplifications:

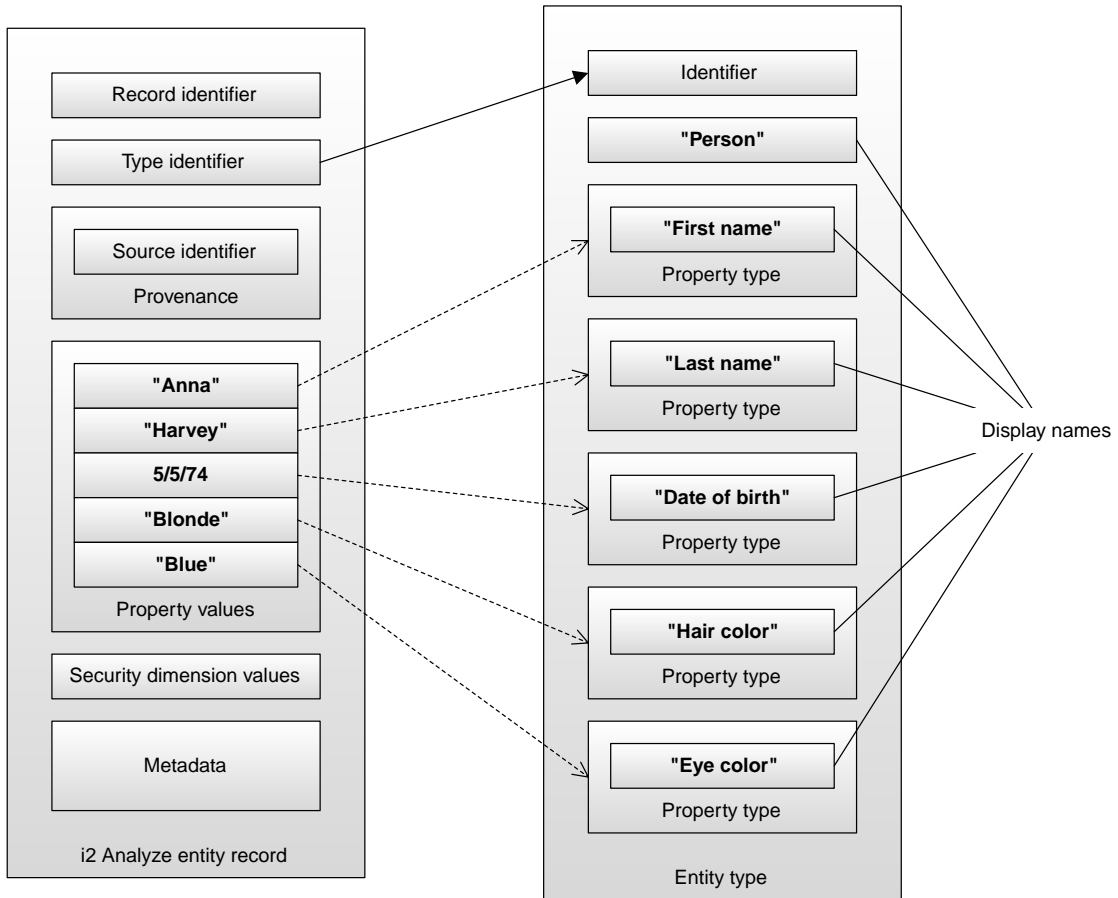
- *Provenance* is about the data sources that contributed to a particular i2 Analyze record. When the property values of an i2 Analyze record represent data from more than one source, that record can have more than one piece of provenance.
- When an i2 Analyze record has more than one piece of provenance, it can contain all the data from all the contributing sources. In that case, the property values that the record presents are derived from the source data.
- *Metadata* includes the following information:
 - Timestamps, which reflect when data in an i2 Analyze record was created or modified
 - *Source references*, which describe the sources that the data in a record came from
 - *Notes*, which users can write to add free-form text commentary to a record

For link records, the metadata also includes information about the strength and direction of the link.

As an example of how to represent a simple entity that contains data from a single source, consider the following information about a person:

Full name Anna Harvey
 Date of birth 5/5/74
 Hair color Blonde
 Eye color Blue

The following diagram shows one way to represent this information as an i2 Analyze record:



Note: An i2 Analyze entity record can contain properties that have any of the property types that the entity type defines. However, one record can contain only one property of each defined type.

The diagram also shows how the property types in the schema only partially determine the contents of an i2 Analyze record. Some of the other contents are due to the security schema, while others still are about identification:

- All i2 Analyze records contain security dimension values, which i2 Analyze uses to determine the access level to the record that a particular user has.
- When they enter the system (through ingestion to the Information Store, or through Analyst's Notebook Premium), i2 Analyze records receive a universally unique *record identifier*. This identifier

is permanent for the lifetime of the record. If they have the necessary access level, any user of the system can use the record identifier to refer to a particular record.

- i2 Analyze records that began life in an external data source contain one or more pieces of provenance. Each piece has a source identifier that references the data for the record in its original source. One record can have provenance from more than one source.

Note: For records in an Information Store that were loaded through ingestion, source identifiers have the additional feature of being unique within the store. These source identifiers are known as *origin identifiers*.

- All i2 Analyze records can contain timestamps in their metadata that specify when source data for the record was created or edited.
- i2 Analyze link records contain an indication of their direction. i2 Analyze considers links to go 'from' one entity 'to' another. The direction of a link can be with or against that flow, or it can run in both directions or none.

When i2 Analyze records are stored in an Information Store, they contain a few extra pieces of data:

- All i2 Analyze records retain timestamps in their metadata for when they were first created or uploaded to the Information Store, for when they were most recently uploaded, and for when they were last updated.
- All i2 Analyze records can contain a *correlation identifier*. If two records have the same correlation identifier, the platform considers that they represent the same the real-world object and might merge them together.

Your data sources are likely to contain some, but not all, of the data that i2 Analyze records require. To enable an Information Store to ingest your data, or to develop a connector for the i2 Connect gateway, or to write an import specification, you must provide the extra information to i2 Analyze.

Identifiers in i2 Analyze records

i2 Analyze records make extensive use of identifiers. Records use identifiers to refer to their type in an i2 Analyze schema, to their original source data, and to other records in ELP relationships. Preparing data for compatibility with i2 Analyze often involves creating or providing the identifiers that form the basis for the reference mechanisms.

Type identifiers

Every i2 Analyze record contains a type identifier, which is a reference to one of the entity types or link types that a schema defines. When you create an ingestion mapping file, an import specification, or a connector, you must arrange for each incoming record to receive the identifier of a type definition.

Every i2 Analyze *link* record contains two further type identifiers, which are references to the entity types of the records at the ends of the link. You must arrange for incoming link records to receive these identifiers as well.

This strong typing of records in i2 Analyze is key to the analytical functions that the platform provides. It allows users to consider not only the existence of relationships between records, but also the nature of those relationships. Schemas define exactly what relationships to allow between record types, and i2 Analyze enforces those rules during record creation.

Record identifiers

i2 Analyze records are created when you ingest data into the Information Store, or when a user creates an item that contains an i2 Analyze record on the chart surface by:

- Importing data through an import specification

- Adding the results of an operation against an external source
- Using an i2 Analyze palette in Analyst's Notebook Premium

At creation, every i2 Analyze record automatically receives a universally unique record identifier that is permanent for the lifetime of that record. Users and administrators of an i2 Analyze deployment can use the record identifier as a convenient way to refer to a record in features such as text search and the Investigate Add-On.

Source identifiers

The role of a source identifier is to refer to the data for a record reproducibly in its original source. If a record represents data from several sources, then it contains several source identifiers. The nature of a source identifier depends on the source and the record creation method, and sometimes on whether the record is a link or an entity.

When you write [ingestion mappings](#) or develop [connectors](#) for the i2 Connect gateway, you are responsible for providing the identifying information. For example, if the original source is a relational database, then entity data is likely to have ready-made source identifiers: table names and primary keys. Link data can also have ready-made source identifiers, but it might not, especially if the relationship that the link represents exists only as a foreign key.

If the source of a record is a text file, then the file name might form part of the source identifier, along with some reference to the data within the file.

Note: Source identifiers are not displayed to end users, but they are a part of the data that records contain. Avoid including sensitive information such as passwords, or configuration detail such as IP addresses. Assume that any information you use as part of a source identifier might be read by users of the system.

Origin identifiers

In general, source identifiers are not certain to be unique within a deployment of i2 Analyze. Several users might independently retrieve the same data from an external source, resulting in several records with the same source identifier. However, when you *ingest* data into the Information Store, i2 Analyze compares the incoming source identifier with existing records. If it finds a match, i2 Analyze updates a record instead of creating one.

The source identifiers that records receive during ingestion therefore *are* unique within i2 Analyze, and they have a special name in this context. They are called *origin identifiers*.

Correlation identifiers

The purpose of a correlation identifier is to indicate that the data in an i2 Analyze record pertains to a particular real-world object. As a result, correlation identifiers are usually related to property values rather than other identifiers. (For example, two Person records from different sources that contain the same social security number are likely to contain data about the same real person.) When two records have the same correlation identifier, they represent the same real-world object, and are candidates to be merged.

When you ingest data into the Information Store, you can provide a correlation identifier for each incoming record. For more information about correlation identifiers and how to create them, see [Correlation identifiers](#).

Security of i2 Analyze records

i2 Analyze records are subject to the i2 Analyze security rules. The security schema defines the security model for your i2 Analyze deployment, and every i2 Analyze record must have at least one value from each security dimension in the schema.

When a user runs a query, i2 Analyze looks up which groups the user belongs to, and determines their security permissions. Then, it compares their security permissions to the security dimension values of the records in the query results. In this way, i2 Analyze calculates which records the user has access to.

If your deployment of i2 Analyze includes an Information Store that you populate through ingestion, then you must add security information to the data during that process. Each time the process runs, you can specify which security dimension values the incoming records receive:

- If you decide that all the data from a given external source must have the same security settings, you can specify the same dimension values for records of all types.
- Alternatively, you can dictate that all the records of a particular entity type or link type receive the same security dimension values.
- You can also configure the process so that individual records receive security dimension values that you specify or determine programmatically.

Note: In this version of i2 Analyze, you can change the security dimension values of an ingested record only by ingesting it again with a different set of values.

All other i2 Analyze records receive dimension values when users create them in Analyst's Notebook Premium by importing them, or searching an external source, or by using an i2 Analyze palette. These records start with default values that users can edit in the same way that they can edit property values.

Storing charts in i2 Analyze

In addition to storing records in the Information Store, i2 Analyze can provide a shared Chart Store for Analyst's Notebook charts. Users of Analyst's Notebook Premium can search for, download, edit, and upload charts in the i2 Analyze Chart Store.

To store charts, i2 Analyze uses a similar model to the one that it uses for storing records in the Information Store. Charts in the Chart Store benefit from features such as timestamps, notes, and source references. They also use the same security model as the rest of the system, so user access to charts is controlled in the same way as user access to records.

Extensions to the record model mean that the Chart Store indexes the *contents* of a chart for searching, in addition to its notes and properties. It also keeps an image of the chart's appearance when it was uploaded. Furthermore, the Chart Store retains a version history and performs basic version control operations for its contents.

Note: At this version of i2 Analyze, the Chart Store treats charts as stand-alone entities and does not model connections between them.

To learn more about the information in the Chart Store and what tasks an administrator of i2 Analyze can use it for, see [Retrieving chart metadata](#).

Security model

All data in i2 Analyze can be secured so that only the users who are supposed to interact with it are able to do so. Using the i2 Analyze security model, you can decide what access users have to records and features, based on their membership of user groups.

In i2 Analyze, all users are members of one or more groups. For example, there might be a group of "administrator" users. There might be separate groups of users for each operational team in your organization. There might be a group of users with higher security clearance than others. The assignment of users to groups is handled at login.

Just as users of i2 Analyze are categorized, so too are records, according to a range of deployment-specific criteria. For example, records might be categorized according to the nature of the information they contain, or how sensitive that information is.

To make sure that users see only the records that they are allowed to see, every deployment of i2 Analyze has a security schema. The security schema defines the categories into which records must be placed, and the relationships that determine what access the users in a particular group get to records in a particular category.

In other words, the i2 Analyze security schema allows you to create rules that say things like, "Users with low security clearance cannot see sensitive records," or "Users in Team A can see only records whose source was signals intelligence." i2 Analyze then combines such rules predictably, on a per-record and per-user basis.

Important: Orthogonal to this security model, i2 Analyze supports blanket controls over the visibility of records with particular types. You can specify that only certain groups of users can see records of a specific type, and that all records of that type are invisible to all other users, regardless of security schema categories. For more information about this functionality, see [Item type security](#).

i2 Analyze security dimensions

In the i2 Analyze security model, a *security dimension* is a way to categorize a record, with the aim of using its category to determine whether particular users are allowed to view or modify it. The available security dimensions in a deployment of i2 Analyze are specific to that deployment, and they are defined in its *security schema*.

A deployment of i2 Analyze might need several different ways to categorize records:

- Records might be categorized by their security classifications
- Records might be categorized by the type of intelligence that produced them
- Records might be categorized by the operational teams who are allowed to access them

As a result, the deployment requires several security dimensions. Each dimension contains a set of values that records can have in order to classify them within that dimension.

To continue the example, the three dimensions might contain values as follows:

Security classification

Top Secret, Secret, Confidential, Restricted

Intelligence type

Human Informant, Open Source

Operational team

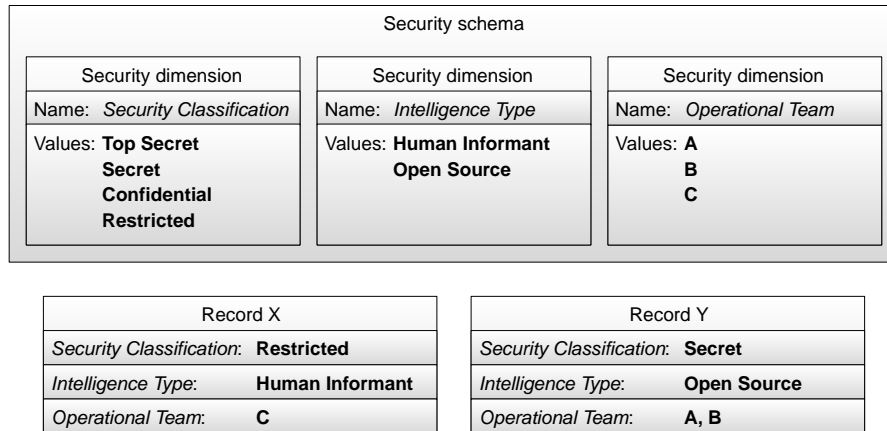
A, B, C

As a result of these definitions, for example, it is possible to mark a record as containing confidential information derived from a human informant, to be available to users in Team B.

In some dimensions (such as security classification), the possible values form a sequence from which each record takes a single value. In these cases, the values act as levels, where each value supersedes all the values after it. If a record is "Top Secret", it cannot be "Restricted" at the same time.

In dimensions such as operational team, where the values do not form a sequence, records can take one or more values. A record might be available to users in Team B and Team C.

Every record in an i2 Analyze deployment must have at least one value from each of the security dimensions in that deployment. There is no such thing as an "optional" dimension. For example:



There are no restrictions on the numbers of dimensions or values that a security schema can define. Keep in mind, though, that the more dimensions there are, the more complicated it becomes to maintain the security schema.

i2 Analyze security permissions

In i2 Analyze, security permissions provide the link between the security dimension values that a record has, and what users in a particular group are allowed to do with that record. The platform calculates the access rights of users according to the permissions of the user groups to which they belong.

The result of the calculation that i2 Analyze performs is that for any record, a user receives one of three security access levels:

None

The user has no access to the record. The user cannot examine the record data, or even know that the record exists.

Read only

The user has read-only access to the record and its data.

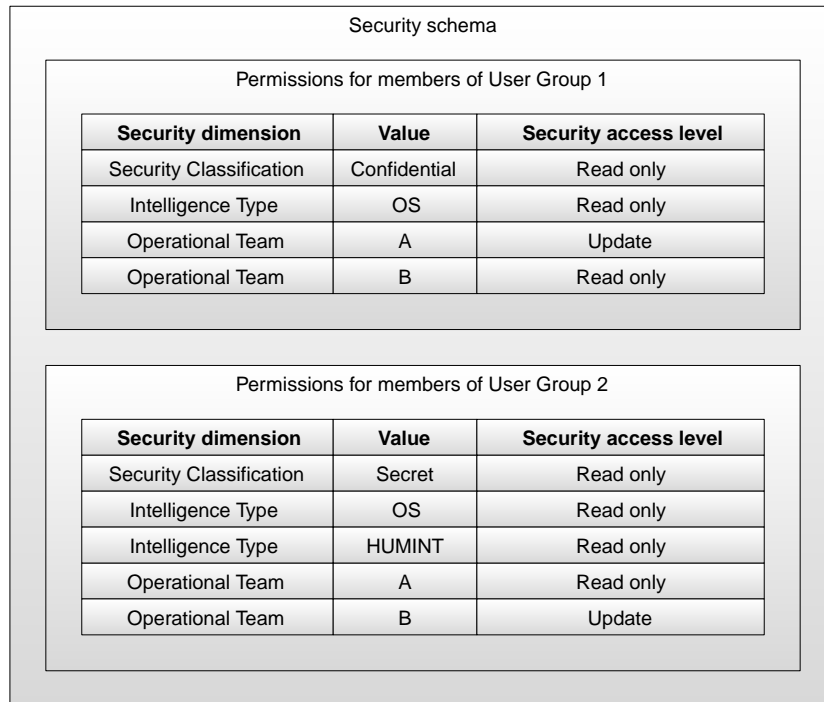
Update

The user can read, modify, and delete the record and its data.

In an i2 Analyze security schema, the set of security permissions for a user group defines mappings from dimension values to access levels. Users receive the security access levels that their user group indicates for the dimension values of a record.

For example, a dimension value might mark a record as containing open source information, and a security permission might state that members of a certain user group have the "Update" access level on records with that dimension value. In that case, a user in that group receives the "Update" access level on that record.

In practice, when a user is a member of several user groups, or a record has multiple dimension values, it is possible for a user to receive several security access levels from different security permissions. In these circumstances, i2 Analyze computes a single security access level from all the contributors.



It is not compulsory for a set of permissions for a user group to provide a security access level for every value of every dimension. Any dimension value that does not appear in a set of permissions receives a default security access level, according to a set of rules:

- For an unordered dimension, a dimension value that does not appear in the permissions receives the "None" level.
- For an ordered dimension:
 - If the unspecified value comes after a dimension value that does appear, then the unspecified value receives the same level as the specified value.
 - If the unspecified value comes before a dimension value that does appear, then the unspecified value receives the "None" level.

For example, if a particular set of permissions associates the "Read only" access level with "Restricted" records (and makes no other setting), then the default access level for "Confidential" records is "None". However, if the permissions associate the "Read only" access level with "Confidential" records instead, then users in the same group receive that access level for "Restricted" records as well.

An i2 Analyze system administrator must arrange the security schema so that all users can receive a security access level that is not "None" for at least one value in every dimension.

Security model example

At any moment, a user has one security access level for each record in i2 Analyze. The platform calculates this level according to a consistent set of rules.

The process for determining a security access level involves examining security permissions within and across dimensions. The platform does the job in three steps:

1. Bring together the permissions for all the user groups of which the user is a member.
2. Use the permissions to determine all the security access levels that the user receives for each dimension value that the record has. Take the least restrictive level in each case.
3. Examine all of these "least restrictive" dimension-specific security access levels, and take the most restrictive.

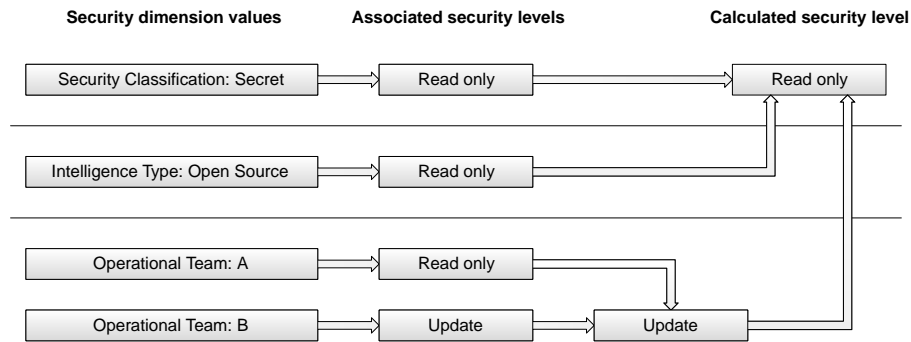
For example, consider the following record, which has one value for each of two security dimensions, and two values for a third.

Record Y	
<i>Security Classification:</i>	Secret
<i>Intelligence Type:</i>	Open Source
<i>Operational Team:</i>	A, B

Then, consider a user in a group that has the following security permissions. (It does not matter whether the permissions are due to one user group or several.)

Security permissions		
Security dimension	Value	Security access level
Security Classification	Secret	Read only
Intelligence Type	OS	Read only
Intelligence Type	HUMINT	Read only
Operational Team	A	Read only
Operational Team	B	Update

The following diagram then represents the process for determining the security access level of the user for this record.



The record has two values in the "Operational Team" dimension that map to different access levels for this user. At this stage in the calculation, the less restrictive access level ("Update") is taken. However, the values from the "Security Classification" and "Intelligence Type" dimensions both map to the "Read only" access level. The final part of the calculation takes the most restrictive level, and the user therefore has the "Read only" access level on this record.

Security architecture

The i2 Analyze security architecture supports the behavior that the i2 Analyze security model requires. Any part of the i2 Analyze application can interact with the security architecture to determine what rights the current user has for the operation that they want to perform.

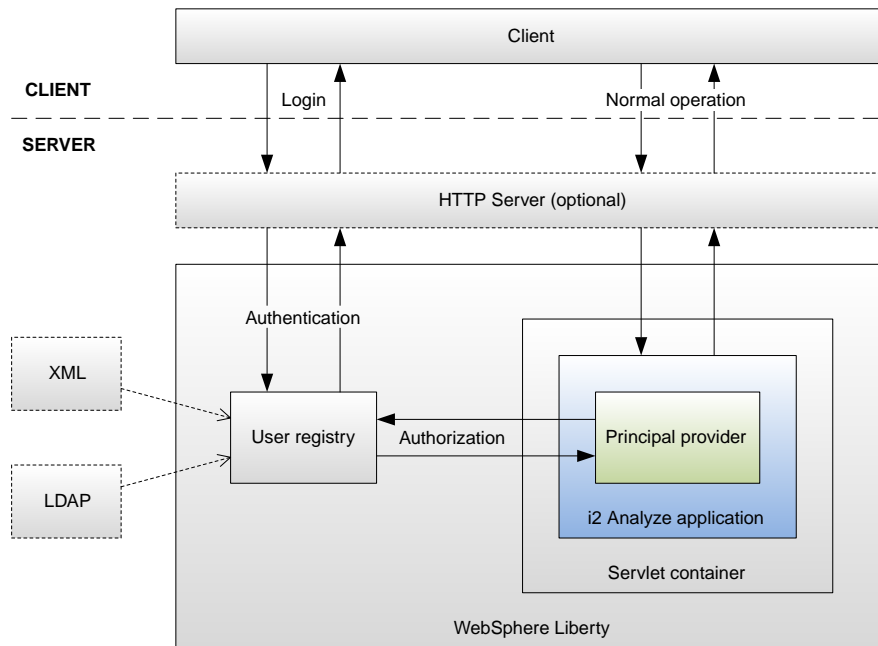
i2 Analyze authenticates users through a choice of technologies, and determines their access level for every record that it manages. The i2 Analyze security model bases its behavior on the interaction between the security dimension values that records have, and the security permissions that user groups convey.

- Records in i2 Analyze are categorized by receiving values from security dimensions. The values that a record has from different security dimensions affect whether users can view or edit that record.
- Security permissions apply to groups of users. On a per-group basis, they associate security access levels with particular dimension values. User group membership is often decided by the team membership or the security clearance of the users that they contain.

The components of an i2 Analyze deployment interact with the security architecture in the following ways:

- At login, WebSphere Liberty requires clients to authenticate before they can interact with i2 Analyze. On successful authentication, the client receives a Lightweight Third-Party Authentication (LTPA) token in a cookie.
- During normal operation, the client passes the cookie back to the i2 Analyze application, which enforces data access rights.

The following diagram shows how security works in a typical i2 Analyze deployment:



In a standard i2 Analyze deployment, authentication and authorization take place as follows:

- Authentication between a client and Liberty uses standard HTTP authentication methods such as basic authentication or form-based authentication.
- Liberty uses its user registry to verify the credentials that the user supplies through the client. The user registry is a service that provides access to user and group information. It can be held in an XML file, one or more LDAP registries, or in any similar store that Liberty can use.
- To authorize users of a client to access records, the i2 Analyze application communicates with the Liberty user registry to retrieve information about the current user's membership of groups. A principal provider then maps the retrieved information to group permissions that are defined in the security permissions section of the i2 Analyze security schema. This mapping is deployment-specific because the security schema is deployment-specific.
- Code in the i2 Analyze application compares the permissions of the current user with the security dimension values of records, to determine what access levels the user receives for each record.

The technologies in the diagram are not fixed. For example, it is possible to use any supported store for the user registry. The requirements are as follows:

- The i2 Analyze application must be able to derive information about a user from the credentials they present.
- A (potentially) deployment-specific module must map user information onto membership of the groups that are named in the security permissions section of the i2 Analyze security schema.

If an implementation of the security architecture fulfills these requirements, then it is suitable for use in an i2 Analyze deployment.

Supplied security implementation

One of the requirements for a deployment of i2 Analyze is a principal provider, which is the mechanism through which the users in an organization are mapped to the user groups in the security

schema. When a deployment environment uses WebSphere Liberty for user authentication, the i2 Analyze deployment toolkit contains a production-quality class that might be an appropriate solution.

The `WebSphereDynamicAccessRoleBasedPrincipalProvider` class from the deployment toolkit performs a direct mapping from the names of user groups in Liberty to the names of user groups in the security schema. When the user is a member of a Liberty group, they receive access levels in accordance with the contents of corresponding `<GroupPermissions>` elements in the i2 Analyze security schema.

The i2 Analyze deployment toolkit can provide an example security schema and Liberty user registry that contain correlating group names and dimension values. These files are suitable for use in development environments, but not in production.

Logging and auditing

i2 Analyze provides mechanisms for logging two types of information that the system generates during normal execution. You can control what information is sent to the system logs, and audit the commands that users invoke.

System logging

The components that make up the i2 Analyze server all contain instrumentation that sends information about the health of the system to log files or the console. You can control the locations of the log files, and the volume of information that the system sends, by editing the `log4j.properties` files in the deployment toolkit.

The information that i2 Analyze can log through this mechanism includes detail about warnings and errors that users see in their client software, and incremental status reports about long-running processes such as ingestion.

For more information about system logging, see the deployment and configuration guides for i2 Analyze, or the Apache Log4j website.

User activity logging

When a user runs an authenticated command against any of its services, i2 Analyze can record information about the user who ran the command, and full details of the command that they ran. For example, you might use this functionality to audit the frequency with which different users make requests for the data that i2 Analyze manages, or to track searches with particular patterns. i2 Analyze handles user activity logging for the i2 Connect gateway separately from the Information Store and the Chart Store.

Note: Depending on the volume of data, enabling user activity logging might affect the performance of i2 Analyze.

Information Store and Chart Store

i2 Analyze supports user activity logging for all of the main analysis operations against the Information Store and (where relevant) the Chart Store. For example, you can configure separate logging (or no logging at all) for search, expand, and find path operations. You can also arrange for logging to occur when records and charts are created or modified.

To audit user activity, including activity due to Analyst's Notebook Premium and the Investigate Add-On, you write a class that implements the `IAuditLogger` interface and specify it in the `ApolloServerSettingsMandatory.properties` file in the deployment toolkit.

At startup, i2 Analyze calls `IAuditLogger` to discover what activities to log information about. Later, it calls again with information such as the time of the activity, the name and security clearance of the user, and the parameters that they supplied.

For more information and an example of how to implement `IAuditLogger`, see [IBM i2 Analyze Developer Essentials](#).

i2 Connect gateway

To log operations against external sources through the i2 Connect gateway, i2 Analyze uses the same `IAuditLogger` interface that it uses for the Information Store and chart store. However, all such operations are logged through a single method on the `IAuditLogger` interface.

Deploying i2 Analyze

All deployments of i2 Analyze are different in terms of the functionality they support and the components they employ. They are also different in terms of how you intend them to be used. The process by which you deploy i2 Analyze changes significantly, depending on whether your target is an example or a production environment.

Deployment information

Deployment types

Before you start a deployment of i2 Analyze, choose the type of deployment that you want to create. Use the following information to ensure that you choose the correct type of deployment for your requirements.

Deployment tasks

Creating an example deployment

To understand what i2 Analyze is, and to demonstrate the features of the system, you can create an example deployment.

Creating a production deployment

The process of creating a production deployment is separated into a number of different activities, which you complete in an iterative process. The suggested process involves the creation and retention of several environments, each one focused on different aspects of a production deployment of i2 Analyze.

Troubleshooting and support

i2 Analyze support page

IBM Software Support home page

IBM Security Learning Academy

IBM training

Deployment types

Before you start a deployment of i2 Analyze, choose the type of deployment that you want to create. Use the following information to ensure that you choose the correct type of deployment for your requirements.

Example deployment

You can use an example deployment to learn about i2 Analyze, demonstrate the features of the system, and ensure that any software prerequisites are installed correctly on a single server.

When you create an example deployment, the deployment toolkit populates all of the mandatory configuration settings with default values and deploys the system. The deployment uses an example i2 Analyze schema, security schema, and data. Some configuration settings that are not mandatory for deployment are also populated to demonstrate extra features of the system.

For more information about example deployments, see [“Creating an example deployment” on page 30](#).

Production deployment

A production deployment is available to analysts to complete mission critical analysis on real-world data. When you decide to create a production deployment of i2 Analyze, you must start from a clean installation of i2 Analyze.

The process for creating a production deployment involves a number of different deployment and configuration activities. As part of the process, you must develop an i2 Analyze schema and security schema for your data.

For more information about production deployments, see [“Creating a production deployment”](#) on page 39.

Creating an example deployment

To understand what i2 Analyze is, and to demonstrate the features of the system, you can create an example deployment.

An example deployment uses default values that are provided by the deployment toolkit, and contains a configuration that demonstrates the features provided by i2 Analyze. You can also use an example deployment to verify that any prerequisites are installed correctly.

Creating an example with the Chart Store

An installation of i2 Analyze includes example settings for deploying the server with the Chart Store. This topic describes how to use those settings in an example deployment.

Install i2 Analyze and any software prerequisites. For more information, see [Installing i2 Analyze](#). To deploy the Chart Store, you need either IBM® Db2® or Microsoft SQL Server 2019. You do not need IBM HTTP Server.

If you are using SQL Server, download the Microsoft JDBC Driver 7.4 for SQL Server archive from <https://www.microsoft.com/en-us/download/details.aspx?id=58505>. Extract the contents of the download, and locate the `sqljdbc_7.4\enu\mssql-jdbc-7.4.1.jre11.jar` file.

Important: For a deployment of Analyst's Notebook Premium to a small workgroup, an example deployment of i2 Analyze with the Chart Store and custom security settings might offer sufficient performance for use in a production environment. For a larger workgroup, or for a group with a large number of charts, a [production deployment](#) is appropriate.

The following procedure describes how to create an example deployment of i2 Analyze with the Chart Store. The i2 Analyze toolkit contains an example configuration for the deployment. The `deployExample` task generates the default values for the mandatory settings and deploys the platform.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in.

In the example deployment, i2 Analyze runs with an example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is Jenny
- The password is Jenny

The example deployment uses the `chart-storage-schema.xml` schema file, with the associated `chart-storage-schema-charting-schemes.xml` file as the charting scheme.

1. Create the configuration directory:

- a) Navigate to the `\toolkit\examples\configurations\chart-storage` directory.

This directory contains the preconfigured files that you need to deploy a system that uses the Chart Store to store Analyst's Notebook charts.

- b) Copy the configuration directory to the root of the toolkit.

For example, `C:\IBM\i2analyze\toolkit\configuration`.

If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example.

2. Copy the example topology.xml file for SQL Server from the toolkit\configuration\examples\topology\sqlserver to the toolkit\configuration\environment directory. Overwrite the existing topology.xml file in the destination directory.
3. Copy the mssql-jdbc-7.4.1.jre11.jar file that you downloaded to the toolkit\configuration\environment\common\jdbc-drivers directory.

Regardless of your database management system, you must complete the following steps after you create the configuration directory.

4. Specify the credentials to use for the deployment.
 - a) Using a text editor, open the toolkit\configuration\environment\credentials.properties file.
 - b) Enter the user name and password to use with the database.
 - c) Enter a user name and password to use for Solr.
 - d) Enter a password to use to encrypt LTPA tokens.
 - e) Save and close the credentials.properties file.
5. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the topology.xml file.

By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see [Db2 server TCP/IP port numbers](#).

- a) Using an XML editor, open the toolkit\configuration\environment\topology.xml file.
 - b) In the <database> element, set the value of the port-number attribute to 25000.
 - c) Save and close the topology.xml file.
6. Run the setup script to create the example deployment.
 - a) Open a command prompt and navigate to the toolkit\scripts directory.
 - b) To deploy the example, run the following command:

```
setup -t deployExample
```

- c) To start the Liberty server, run the following command:

```
setup -t start
```

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

Web application available (default_host): http://host_name:9082/opal.

Install Analyst's Notebook Premium and connect to your deployment. For more information, see ["Connecting clients"](#) on page 92.

Creating an example with the Information Store

An installation of i2 Analyze includes example settings for deploying the server with the Information Store. This topic describes how to use those settings in an example deployment.

Install i2 Analyze and any software prerequisites. For more information, see [Installing i2 Analyze](#). To deploy the preconfigured examples for the Information Store, you need either IBM Db2 or Microsoft SQL Server 2019. You do not need IBM HTTP Server.

If you are using SQL Server, download the Microsoft JDBC Driver 7.4 for SQL Server archive from <https://www.microsoft.com/en-us/download/details.aspx?id=58505>. Extract the contents of the download, and locate the `sqljdbc_7.4\enu\mssql-jdbc-7.4.1.jre11.jar` file.

The following procedure describes how to create an example deployment of i2 Analyze with the Information Store, which also fulfills the functions of the Chart Store. The i2 Analyze toolkit contains an example configuration for the deployment. The `deployExample` task generates the default values for the mandatory settings and deploys the platform.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in.

In the example deployment, i2 Analyze runs with an example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is Jenny
- The password is Jenny

The example deployment uses the `law-enforcement-schema.xml` schema as the i2 Analyze schema with the associated `law-enforcement-schema-charting-schemas.xml` as the charting scheme.

1. Create the configuration directory:

- a) Navigate to the `\toolkit\examples\configurations\information-store-opal` directory.

This directory contains the preconfigured files that you need to deploy a system that uses the Information Store to store data and Analyst's Notebook charts, and to support the i2 Notebook web client.

- b) Copy the configuration directory to the root of the toolkit.

For example, `C:\IBM\i2analyze\toolkit\configuration`.

If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example.

2. Copy the example `topology.xml` file for SQL Server from the `toolkit\configuration\examples\topology\sqlserver` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.

3. Copy the `mssql-jdbc-7.4.1.jre11.jar` file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.

Regardless of your database management system, you must complete the following steps after you create the configuration directory.

4. Specify the credentials to use for the deployment.

- a) Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.

b) Enter the user name and password to use with the database.

c) Enter a user name and password to use for Solr.

d) Enter a password to use to encrypt LTPA tokens.

e) Save and close the `credentials.properties` file.

5. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the `topology.xml` file.

By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see [Db2 server TCP/IP port numbers](#).

- a) Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

b) In the `<database>` element, set the value of the `port-number` attribute to 25000.

- c) Save and close the `topology.xml` file.
6. Run the setup script to create the example deployment.
- a) Open a command prompt and navigate to the `toolkit\scripts` directory.
- b) To deploy the example, run the following command:

```
setup -t deployExample
```

- c) To start the Liberty server, run the following command:

```
setup -t start
```

7. Optional: To populate your Information Store with the provided example data for the `law-enforcement-schema.xml` schema, run the following command:

```
setup -t ingestExampleData
```

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

Web application available (default_host): `http://host_name:9082/opa1`.

Install Analyst's Notebook Premium or open a web browser and connect to your deployment. For more information, see [“Connecting clients” on page 92](#).

Creating an example with the i2 Connect gateway

An installation of i2 Analyze includes example settings for deploying the server with support for the i2 Connect gateway only. With these settings, the i2 Connect gateway enables Analyst's Notebook Premium users to search for and retrieve data from external data sources, and then to analyze the results on charts.

Install i2 Analyze and any software prerequisites. For more information, see [Installing i2 Analyze](#). You do not need IBM HTTP Server.

Before you create the example deployment, you must download and install Node.js to host the example connector. Download Node.js for your operating system from <https://nodejs.org/en/download/>. You can install Node.js with the default settings.

To use any deployment of i2 Analyze with the i2 Connect gateway, you must obtain or create a connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and an example connector with example data.

The example deployment demonstrates a working i2 Analyze system that can query and retrieve data from an external data source. You can log in with an example user. In the example deployment, i2 Analyze runs with the example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is Jenny
- The password is Jenny

Note: This example does not support the i2 Notebook web client. To give users the option of using the web client, you must also [deploy the Chart Store](#).

1. Create the configuration directory:

- a) Navigate to the `\toolkit\examples\configurations\daod-opa1` directory.

This directory contains the preconfigured files that you need to deploy a system that uses the i2 Connect gateway to connect to an external data source.

- b) Copy the configuration directory to the root of the toolkit.
For example, C:\IBM\i2analyze\toolkit\configuration.
2. Specify the credentials to use for the deployment.
 - a) Using a text editor, open the toolkit\configuration\environment\credentials.properties file.
 - b) Enter the user name and password to use for Solr.
 - c) Enter the password to use to encrypt LTPA tokens.
 - d) Save and close the credentials.properties file.
3. Run the setup script to create the example deployment.
 - a) Open a command prompt and navigate to the toolkit\scripts directory.
 - b) To deploy the example, run the following command:

```
setup -t deployExample
```

4. Install any dependencies and start the server that hosts the example connector.

Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

- a) In a command prompt, navigate to the toolkit\examples\connectors\example-connector directory.
- b) To install the dependencies that are required for the example connector, run the following command:

```
npm install
```

Note: You must be connected to the internet to install the dependencies.

- c) To start the Node.js server, run the following command:

```
npm start
```

5. Start i2 Analyze.

- a) Open a command prompt and navigate to the toolkit\scripts directory.
- b) To start i2 Analyze, run the following command:

```
setup -t start
```

When you start i2 Analyze, the URI that you can use to connect to it from Analyst's Notebook Premium is displayed in the console. For example:

```
Web application available (default_host): http://host_name:9082/opaldao
```

Install Analyst's Notebook Premium and connect to your deployment. For more information, see ["Connecting clients" on page 92](#).

Production deployments of i2 Analyze use client-authenticated SSL communication between i2 Analyze and any connectors. The example deployment does not use it, and so Analyst's Notebook Premium displays a warning to that effect when you open the external searches window. For more information about configuring client-authenticated SSL, see [Client-authenticated Secure Sockets Layer with the i2 Connect gateway](#).

You can create your own connectors to use with the deployment of i2 Analyze. For more information, see [IBM i2 Analyze and the i2 Connect gateway](#).

Creating an example with the Chart Store and the i2 Connect gateway

An installation of i2 Analyze includes example settings for deploying the server with the Chart Store and support for the i2 Connect gateway. With these settings, Analyst's Notebook Premium users can upload charts to the Chart Store, while both they and i2 Notebook web client users can search for and retrieve data from an example external data source through the i2 Connect gateway.

Install i2 Analyze and any software prerequisites. For more information, see [Installing i2 Analyze](#). To deploy the Chart Store, you need either IBM Db2 or Microsoft SQL Server 2019. You do not need IBM HTTP Server.

If you are using SQL Server, download the Microsoft JDBC Driver 7.4 for SQL Server archive from <https://www.microsoft.com/en-us/download/details.aspx?id=58505>. Extract the contents of the download, and locate the `sqljdbc_7.4\enu\mssql-jdbc-7.4.1.jre11.jar` file.

Before you create the example deployment, you must download and install Node.js to host the example connector. Download Node.js for your operating system from: <https://nodejs.org/en/download/>. You can install Node.js with the default settings.

The following procedure describes how to create an example deployment of i2 Analyze with the Chart Store and the i2 Connect gateway. To use any deployment of i2 Analyze with the i2 Connect gateway, you must obtain or create a connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and an example connector with example data. The `deployExample` task generates the default values for the mandatory settings and deploys the platform.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in.

In the example deployment, i2 Analyze runs with the example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is Jenny
- The password is Jenny

The example deployment uses the `chart-storage-schema.xml` schema file, with the associated `chart-storage-schema-charting-schemes.xml` file as the charting scheme.

1. Create the configuration directory:

- a) Navigate to the `\toolkit\examples\configurations\chart-storage-daod` directory.

This directory contains the preconfigured files that you need to deploy a system that uses the i2 Connect gateway to connect to an external data source, and the Chart Store to store Analyst's Notebook charts and support the i2 Notebook web client.

- b) Copy the configuration directory to the root of the toolkit.
For example, `C:\IBM\i2analyze\toolkit\configuration`.

If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example.

2. Copy the example `topology.xml` file for SQL Server from the `toolkit\configuration\examples\topology\sqlserver` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
3. Copy the `mssql-jdbc-7.4.1.jre11.jar` file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.

Regardless of your database management system, you must complete the following steps after you create the configuration directory.

4. Specify the credentials to use for the deployment.
 - a) Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
 - b) Enter the user name and password to use with the database.
 - c) Enter a user name and password to use for Solr.
 - d) Enter a password to use to encrypt LTPA tokens.
 - e) Save and close the `credentials.properties` file.
5. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the `topology.xml` file.

By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see [Db2 server TCP/IP port numbers](#).

- a) Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
 - b) In the `<database>` element, set the value of the `port-number` attribute to 25000.
 - c) Save and close the `topology.xml` file.
6. Run the setup script to create the example deployment.
 - a) Open a command prompt and navigate to the `toolkit\scripts` directory.
 - b) To deploy the example, run the following command:

```
setup -t deployExample
```

7. Install any dependencies and start the server that hosts the example connector.

Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

- a) In a command prompt, navigate to the `toolkit\examples\connectors\example-connector` directory.
 - b) To install the dependencies that are required for the example connector, run the following command:

```
npm install
```

Note: You must be connected to the internet to install the dependencies.

- c) To start the Node.js server, run the following command:

```
npm start
```

8. Start i2 Analyze.

- a) Open a command prompt and navigate to the `toolkit\scripts` directory.
 - b) To start i2 Analyze, run the following command:

```
setup -t start
```

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

```
Web application available (default_host): http://host_name:9082/opa1.
```

Install Analyst's Notebook Premium or open a web browser and connect to your deployment. For more information, see [“Connecting clients” on page 92](#).

Production deployments of i2 Analyze use client-authenticated SSL communication between i2 Analyze and any connectors. The example deployment does not use it, and so Analyst's Notebook Premium displays a warning to that effect when you open the external searches window. For more information about configuring client-authenticated SSL, see [Client-authenticated Secure Sockets Layer with the i2 Connect gateway](#).

You can create your own connectors to use with the deployment of i2 Analyze. For more information, see [IBM i2 Analyze and the i2 Connect gateway](#).

Creating an example with the Information Store and the i2 Connect gateway

An installation of i2 Analyze includes example settings for deploying the server with the Information Store and support for the i2 Connect gateway. Users can access data in the Information Store in the usual manner, and the i2 Connect gateway enables analysts to search for and retrieve data from an example external data source.

Install i2 Analyze and any software prerequisites. For more information, see [Installing i2 Analyze](#). To deploy the preconfigured examples for the Information Store, you need either IBM Db2 or Microsoft SQL Server 2019. You do not need IBM HTTP Server.

If you are using SQL Server, download the Microsoft JDBC Driver 7.4 for SQL Server archive from <https://www.microsoft.com/en-us/download/details.aspx?id=58505>. Extract the contents of the download, and locate the `sqljdbc_7.4\enu\mssql-jdbc-7.4.1.jre11.jar` file.

Before you create the example deployment, you must download and install Node.js to host the example connector. Download Node.js for your operating system from <https://nodejs.org/en/download/>. You can install Node.js with the default settings.

The following procedure describes how to create an example deployment of i2 Analyze with the Information Store, which also fulfills the functions of the Chart Store, and the i2 Connect gateway. To use any deployment of i2 Analyze with the i2 Connect gateway, you must obtain or create a connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and an example connector with example data. The `deployExample` task generates the default values for the mandatory settings and deploys the platform.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in. You can ingest example data into the Information Store and then perform searches and analysis, and use the example connector to query and retrieve data from an external data source.

In the example deployment, i2 Analyze runs with the example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is Jenny
- The password is Jenny

The example deployment uses the `law-enforcement-schema.xml` schema as the i2 Analyze schema with the associated `law-enforcement-schema-charting-schemas.xml` as the charting scheme.

1. Create the configuration directory:

- a) Navigate to the `\toolkit\examples\configurations\information-store-daod-opal` directory.

This directory contains the preconfigured files that you need to deploy a system that uses the i2 Connect gateway to connect to an external data source, and the Information Store to store data and Analyst's Notebook charts, and to support the i2 Notebook web client.

- b) Copy the configuration directory to the root of the toolkit.
For example, C:\IBM\i2analyze\toolkit\configuration.

If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example.

2. Copy the example topology.xml file for SQL Server from the toolkit\configuration\examples\topology\sqlserver to the toolkit\configuration\environment directory. Overwrite the existing topology.xml file in the destination directory.
3. Copy the mssql-jdbc-7.4.1.jre11.jar file that you downloaded to the toolkit\configuration\environment\common\jdbc-drivers directory.

Regardless of your database management system, you must complete the following steps after you create the configuration directory.

4. Specify the credentials to use for the deployment.
 - a) Using a text editor, open the toolkit\configuration\environment\credentials.properties file.
 - b) Enter the user name and password to use with the database.
 - c) Enter a user name and password to use for Solr.
 - d) Enter a password to use to encrypt LTPA tokens.
 - e) Save and close the credentials.properties file.
5. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the topology.xml file.

By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see [Db2 server TCP/IP port numbers](#).

- a) Using an XML editor, open the toolkit\configuration\environment\topology.xml file.
 - b) In the <database> element, set the value of the port-number attribute to 25000.
 - c) Save and close the topology.xml file.
6. Run the setup script to create the example deployment.
 - a) Open a command prompt and navigate to the toolkit\scripts directory.
 - b) To deploy the example, run the following command:

```
setup -t deployExample
```

7. Install any dependencies and start the server that hosts the example connector.

Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

- a) In a command prompt, navigate to the toolkit\examples\connectors\example-connector directory.
 - b) To install the dependencies that are required for the example connector, run the following command:

```
npm install
```

Note: You must be connected to the internet to install the dependencies.

- c) To start the Node.js server, run the following command:

```
npm start
```

8. Start i2 Analyze.

- a) Open a command prompt and navigate to the `toolkit\scripts` directory.
- b) To start i2 Analyze, run the following command:

```
setup -t start
```

9. Optional: To populate your Information Store with the provided example data for the `law-enforcement-schema.xml` schema, run the following command:

```
setup -t ingestExampleData
```

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

Web application available (default_host): `http://host_name:9082/opal`.

Install Analyst's Notebook Premium or open a web browser and connect to your deployment. For more information, see [“Connecting clients”](#) on page 92.

Production deployments of i2 Analyze use client-authenticated SSL communication between i2 Analyze and any connectors. The example deployment does not use it, and so Analyst's Notebook Premium displays a warning to that effect when you open the external searches window. For more information about configuring client-authenticated SSL, see [Client-authenticated Secure Sockets Layer with the i2 Connect gateway](#).

You can create your own connectors to use with the deployment of i2 Analyze. For more information, see [IBM i2 Analyze and the i2 Connect gateway](#).

Creating a production deployment

The process of creating a production deployment is separated into a number of different activities, which you complete in an iterative process. The suggested process involves the creation and retention of several environments, each one focused on different aspects of a production deployment of i2 Analyze.

Planning for production

Before you can start to install and deploy i2 Analyze, you must first understand how i2 Analyze fits into your organization:

- Understand what i2 Analyze is.
- Understand the requirements of the deployment, and align these to a deployment pattern.
- Understand the data and security models of the environment that i2 Analyze is deployed in.

For more information, see the [Understanding](#) section.

Deploying

After you identify the requirements of the deployment, you can start to create the production deployment. The process of deploying i2 Analyze is completed in three phases that are explained in [“Deployment phases and environments”](#) on page 40.

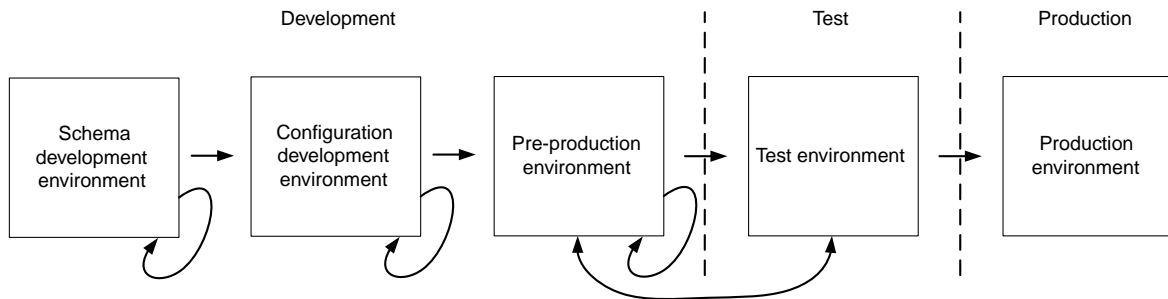
Note:

If your planned deployment of i2 Analyze is to a small workgroup and includes only the Chart Store, an example deployment with the Chart Store and custom security settings might offer sufficient

performance. For more information about example deployments, see [“Creating an example with the Chart Store”](#) on page 30.

Deployment phases and environments

The process of creating a production deployment involves distinct phases that focus on different aspects of an i2 Analyze deployment. You should complete the activities in each phase in distinct environments, which you should retain to refer to later in the process.



There are three phases in the process to deploy i2 Analyze into production.

Development

The *development phase* is where you configure i2 Analyze to meet the requirements of the final deployment. In this phase, you develop the configuration in an iterative process that involves a number of configuration changes and deployments of the system. During this phase, the lifetime of a deployment is short.

Test

The *test phase* is where you deploy i2 Analyze for testing. In the test phase, you deploy i2 Analyze with the configuration from the development phase and perform comprehensive testing of the system.

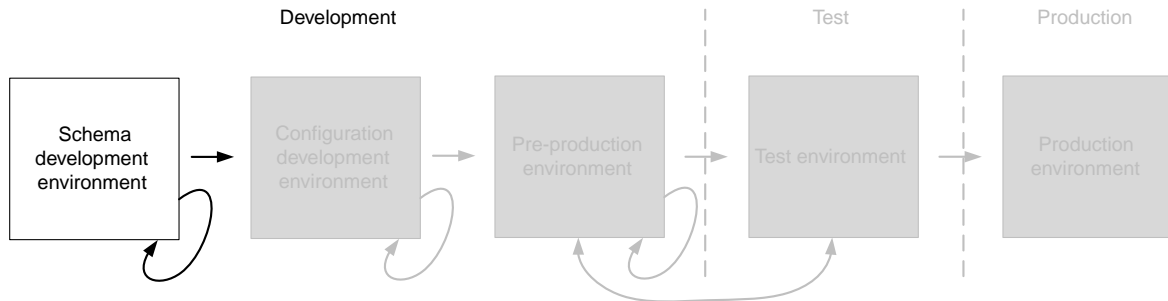
Production

The *production phase* is where you deploy i2 Analyze into production. i2 Analyze is deployed with the configuration that you tested in the test phase. In production, the deployment is fully operational and used by analysts to complete mission critical work.

To start creating your production deployment, complete the instructions in [“Schema development environment”](#) on page 41.

Schema development environment

The first task in the development phase is often to develop the schema and the security schema for the deployment. You can use a *schema development environment*, which features a single-server deployment of i2 Analyze with the i2 Connect gateway to develop the schemas.



In an environment that includes a database, significant changes to the schema or the security schema can be time-consuming. Destructive changes to either schema require you to rebuild the database. The purpose of the schema development environment is to enable rapid iteration. When you apply the schemas that you create here to the Information Store in the configuration development environment, they are less likely to need significant changes.

Creating the schema development environment

In the schema development environment, you first populate the values of some of the mandatory settings for deploying i2 Analyze. After you deploy i2 Analyze for the first time, you can then develop the schemas to meet your requirements.

Depending on the terms of your license and the needs of your organization, a deployment of i2 Analyze might include the Chart Store alone, or the Information Store alone, or the i2 Connect gateway. The i2 Connect gateway can itself appear alone, or in combination with the Chart Store or the Information Store.

- If your target deployment includes only the Chart Store, then *you do not need to create a schema development environment*. Instead, you can go directly to the [configuration development environment](#).
- If your target deployment does not include the Information Store but does include the i2 Connect gateway, then the schema development environment is a convenient place to develop a gateway schema. If you do not intend to use a gateway schema, the environment still provides support for prototyping a security schema.
- If your target deployment includes the Information Store, then you must complete the steps in this and subsequent topics to develop the schema that will become your Information Store schema.

The schema development environment features an instance of i2 Analyze with the i2 Connect gateway; there is no Information Store or Chart Store. By using a deployment without a database, you can quickly prototype changes to the schema and the security schema, and then visualize the changes in Analyst's Notebook Premium.

1. Install any prerequisite software to prepare your server for the schema development environment.
 - For the schema development environment, prepare your server for the [i2 Connect deployment topology](#).

- Install Analyst's Notebook Premium to connect to your deployment and IBM i2 Analyze Schema Designer to edit your schema files. For more information, see [Installing IBM i2 Analyst's Notebook Premium](#).
2. In the deployment toolkit that you installed, copy the `toolkit\examples\configurations\daod-opal\configuration` directory to the `toolkit` directory.
 3. Enter a user name and password to use with the Solr index and a password to encrypt the LTPA tokens in the `configuration\environment\credentials.properties` file.
For more information about credentials in i2 Analyze, see [“Specifying the deployment credentials” on page 67](#).
 4. In a command prompt, navigate to the `toolkit\scripts` directory, and run the following command to populate some mandatory settings with default values:

```
setup -t generateDefaults
```

The `environment.properties` and `topology.xml` are modified by this toolkit task. For more information about the default values that are provided, see [Configuration files reference](#).

5. In i2 Analyze Schema Designer, either create a new schema or open one of the examples to modify.

For information about creating or modifying schema files, see [Creating schemas](#) and [Charting schemes](#).

Example schema files are located in subdirectories of the `configuration\examples\` directory. For more information, see [“Example schemas” on page 59](#).

Note:

If your planned deployment does not include the Information Store and you do not intend to use a gateway schema, your choice of example here is not important. You just need to set up a valid development environment.

- a) Save the initial version of your schema in the `configuration\fragments\common\WEB-INF\classes` directory.
A charting scheme file is saved in the same location when you save the schema.
 - b) Keep the schema file open in Schema Designer so that you can make more modifications after you deploy i2 Analyze.
6. Copy the example security schema to the `configuration\fragments\common\WEB-INF\classes` directory.
The example security schema file is located in the `configuration\examples\security-schema` directory. For more information, see [“Example schemas” on page 59](#).
 7. In the `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties` file, set the values of the following settings to the file names of your schema, charting scheme, and security schema:
 - `Gateway.External.SchemaResource`
 - `Gateway.External.ChartingSchemesResource`
 - `DynamicSecuritySchemaResource`For example, `Gateway.External.SchemaResource=custom-schema.xml`
 8. Deploy i2 Analyze:

```
setup -t deploy
```


9. Create an example user that you can use to log in:

```
setup -t ensureExampleUserRegistry
```

The user has the user name 'Jenny' and the password 'Jenny'.

10. Start i2 Analyze:

```
setup -t startLiberty
```

When you start i2 Analyze, the URI that you can use to connect to it from Analyst's Notebook Premium is displayed in the console. For example:

```
Web application available (default_host): http://host_name:9082/opaldao
```

11. Connect to your deployment by using Analyst's Notebook Premium. Log in with the example 'Jenny' user.

For more information, see [Connecting IBM i2 Analyst's Notebook Premium to IBM i2 Analyze](#).

12. In Analyst's Notebook Premium, create items on the chart to visualize the i2 Analyze schema by using the **Gateway** palette.

After you deploy i2 Analyze with the initial schema files, you can develop them for your own data requirements:

- [“Updating the schema and the charting scheme” on page 43](#)
- [“Updating the security schema” on page 44](#)

After you develop your schema files, you move to the configuration development environment. In the configuration development environment, you deploy i2 Analyze with the schemas that you developed in your schema development environment.

You can choose whether to retain your schema development environment. If you need to make small changes to your schemas later, you can do this in your configuration development environment. If you need to make more substantial changes, you might create another schema development environment.

When you have finalized your schemas, you can move to the next environment. For more information, see [“Configuration development environment” on page 45](#).

Updating the schema and the charting scheme

In the schema development environment, you can quickly deploy your changes to the schema and the charting scheme. Use this environment to develop the schema and charting scheme for your production deployment.

Develop a schema and an associated charting scheme to meet your data requirements. Then update your schema development environment with the modified schema and charting scheme.

1. In i2 Analyze Schema Designer, modify the schema that is deployed in your schema development environment.

For information about creating or modifying your schema files, see [Creating schemas and Charting schemes](#).

After you modify your schema, update the deployment with your changes.

2. Update and redeploy the system:

```
setup -t updateConnectorsConfiguration
```

3. Start i2 Analyze:

```
setup -t restartLiberty
```

4. Test the changes to the schema and charting scheme by connecting to your deployment in Analyst's Notebook Premium and modeling representative data on the chart by using the **Gateway** palette.

Repeat this process until the schema and the charting scheme meet your requirements. When your schema development is complete, store your schema and charting scheme files in a version control system.

This is not the final chance to modify the schema, but it should now contain all the entity and link types that you require, and most of the property types.

When you are satisfied with the schema, develop the security schema for the deployment. For more information, see [“Updating the security schema” on page 44](#).

Updating the security schema

In the schema development environment, you can quickly make and deploy changes to the i2 Analyze security schema. Use this environment to develop the security schema for your production deployment.

Develop a security schema to meet your security requirements. Then update your schema development environment with the modified security schema.

1. In an XML editor, either create a new security schema or open the one that is deployed in the schema development environment.

For information about creating or modifying a security schema file for i2 Analyze, see [The i2 Analyze Security schema](#).

- a) Save your security schema in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.
- b) Ensure that your security schema file is specified in `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties`.

After you modify your security schema, update the deployment with your changes.

2. Update and redeploy the system:

```
setup -t updateSecuritySchema
setup -t deployLiberty
```

3. Start i2 Analyze:

```
setup -t restartLiberty
```

4. If you changed the names of the user groups in the security schema, update the basic user registry to match the new names.

For more information, see [Configuring the Liberty user registry](#).

5. Test the changes to the security schema by connecting to the deployment in Analyst's Notebook Premium as different users and changing the security permissions on records that you create.

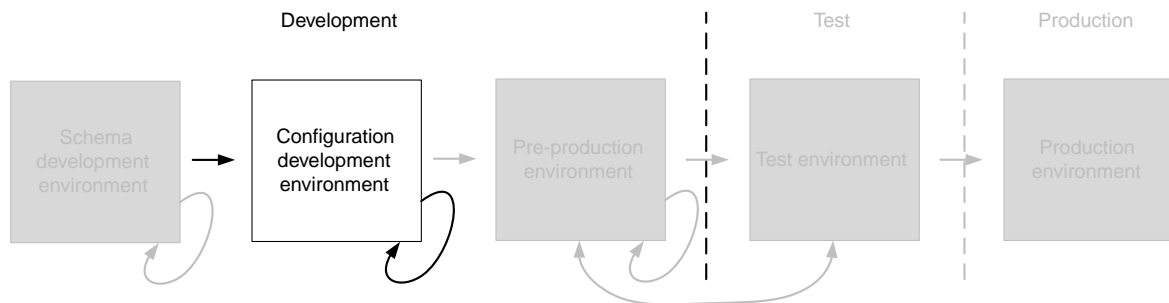
Repeat this process until your security schema meets your requirements. When your security schema development is complete, store your security schema and Liberty user registry files in a version control system.

This is not the final time that you can modify the security schema, but you should aim to have most of the security dimensions and dimension values defined.

After you finish developing your schema files, you can move to the next environment. For more information, see [“Configuration development environment”](#) on page 45.

Configuration development environment

In the configuration development environment, you configure i2 Analyze to meet your requirements. The aspects of i2 Analyze that you might configure include how your data is added to the system and how analysts interact with that data.



After you create your schemas, you can use a *configuration development environment* to configure i2 Analyze in a single server environment. In the configuration development environment, you deploy i2 Analyze with the same data store as your intended production system.

In this deployment of i2 Analyze, you use any schema files that you previously created. If your intended production system includes the Information Store and you have not created your schema files, complete the instructions in [“Schema development environment”](#) on page 41.

Creating the configuration development environment

When you create the configuration development deployment, you deploy i2 Analyze on a single server in the same deployment pattern as your intended production system.

If you intend your production deployment to contain only the i2 Connect gateway, and to be accessed only through Analyst's Notebook Premium, you can use your schema development environment as your configuration development environment. In this scenario, you can move to [develop the i2 Analyze configuration](#).

If your intended production deployment contains the Information Store, ensure that you have access to the following files from the schema development environment so that you can copy them to the configuration development environment:

- The Information Store schema
- The associated charting scheme
- The security schema
- The user registry, `user-registry.xml`

If you did not create a schema development environment because you intend your production environment to contain the Chart Store, you must retrieve equivalent files from the deployment toolkit:

- The Chart Store schema, `chart-storage-schema.xml`
- The associated charting scheme, `chart-storage-schema-charting-schemes.xml`
- The example security schema, `example-dynamic-security-schema.xml`

For deployments that include the Chart Store, the following procedure explains how to create an example user registry when it becomes necessary to do so.

1. Install any prerequisite software to prepare your server for the configuration development environment.
 - For the configuration development environment, prepare your server for the [Single server deployment topology](#) with the same components and database management system as in your intended production system.
2. Create the configuration directory in the i2 Analyze deployment toolkit.
 - a) Navigate to the `toolkit\examples\configurations` directory where you installed i2 Analyze.
 - b) Copy the configuration directory from your chosen base configuration to the root of the toolkit.
For example, copy the `toolkit\examples\configurations\information-store-opal\configuration` directory to the toolkit directory.
For more information about the different base configurations, see [“The base configurations” on page 57](#).
3. If you are using SQL Server as your database management system, copy the `example topology.xml` file from `configuration\examples\topology\sqlserver` to the `configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
4. If you are using IBM HTTP Server, in the `topology.xml` file set the value of the `http-server-host` attribute to `true`.

To allow the deployment toolkit to create and modify components of i2 Analyze, you provide user names and passwords for each of the components in your configuration.

5. Specify the user names and passwords to use for the deployment in the `toolkit\configuration\environment\credentials.properties` file.
For more information about credentials in i2 Analyze, see [“Specifying the deployment credentials” on page 67](#).
6. Copy the JDBC driver to use with the deployment to the `configuration\environment\common\jdbc-drivers` directory.
For more information, see [Specifying the JDBC driver](#).
You do not need to provide a JDBC driver for deployment that contains only the i2 Connect gateway.
7. Use the deployment toolkit to populate some of the mandatory settings with default values:

```
setup -t generateDefaults
```

The `environment.properties` and `topology.xml` are modified by this toolkit task. For more information about the default values that are provided, see [Configuration files reference](#).

8. Specify the i2 Analyze schema, charting scheme, and security schema that you previously prepared.

- a) Copy your i2 Analyze schema, charting scheme, and security schema files to the configuration\fragments\common\WEB-INF\classes directory.
- b) In the configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties file, set the file names of the schema, charting scheme, and security schema.
 - If your deployment includes the Information Store or the Chart Store, set SchemaResource and ChartingSchemesResource.
 - If your deployment includes the i2 Connect gateway and you have developed a gateway schema, set Gateway.External.SchemaResource and Gateway.External.ChartingSchemesResource
 - In both cases, set DynamicSecuritySchemaResource

9. Deploy i2 Analyze:

```
setup -t deploy
```

10. If you created a schema development environment, copy the user registry file from that environment to the deploy\wlp\usr\shared\config directory in the new environment.

If you did not create a schema development environment, execute the following command to create an example user that you can use to log in:

```
setup -t ensureExampleUserRegistry
```

The user has the user name 'Jenny' and the password 'Jenny'.

11. Start i2 Analyze:

```
setup -t start
```

The URI that users must specify is displayed in the console. For example:

```
Web application available (default_host): http://host_name:9082/opa1/
```

After you deploy i2 Analyze, you can begin [“Developing the i2 Analyze configuration”](#) on page 47.

Developing the i2 Analyze configuration

At this stage of the production deployment process, you can use the configuration development environment to define how the i2 Analyze application behaves specifically for your organization.

Developing the configuration is separated into two parts: enabling i2 Analyze to work with your data, and defining how analysts interact with your data.

The i2 Analyze deployment in the configuration development environment is not your final deployment. When you are configuring the deployment, use small amounts of data with a few users to ensure that the processes are functional.

Depending on the composition of your deployment, there are three methods that you can use to get your data into i2 Analyze for analysis.

1. To develop the process for ingesting data into the Information Store, refer to [Ingesting data into the Information Store](#).
2. To develop connections to external data sources, refer to [Connecting to external data sources](#).
3. Ensure that analysts can import and create representative records in Analyst's Notebook Premium. Then, if required, upload records to the Information Store. For more information, see [Import data and Create i2 Analyze chart items](#).

When you develop the process to get your data into i2 Analyze, you might realize that your schema or security schema are not correct for your data. You can update the deployed schemas to better represent your data and security model. Some changes require you to remove and recreate the underlying database.

4. To update your deployed Information Store or Chart Store schema, refer to [Changing the schema](#).
5. To update your deployed security schema, refer to [Configuring the security schema](#).

After you develop the mechanisms for making data available to analysts, you can configure how analysts interact with the data when they use the system. The list of things that you can configure includes:

6. To configure which features or types of commands analysts can access, refer to [Controlling access to features](#).
7. To configure how analysts search for information, and the options that are available to them, refer to [Configuring search](#).
8. To configure how analysts can identify matching records, refer to [Configuring matching](#).
9. To configure user security, refer to [Configure user authentication and authorization](#).

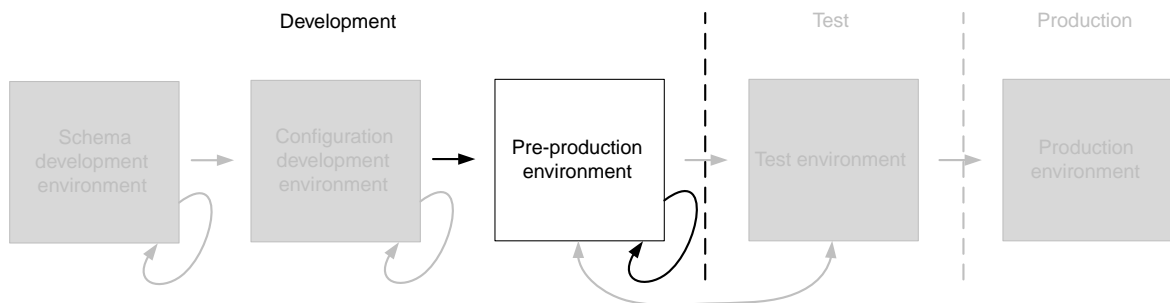
For more information about the configuration changes that you can make, see [Configuring i2 Analyze](#).

After you configure your deployment sufficiently in the single-server environment, you can move to another environment that is more representative of the production deployment. Keep your configuration development environment in place so that you can access the configuration directory in later phases of the production process, and return to it to make further configuration changes.

Next, create the [“Pre-production environment”](#) on page 48.

Pre-production environment

Your target deployment topology is probably different from the single-server configuration development environment. In the pre-production environment, deploy i2 Analyze in the deployment topology that matches your intended production deployment.



After you develop the i2 Analyze configuration, you can use a more representative *pre-production environment* for aspects of the configuration that rely on environment-specific variables. In this environment, deploy i2 Analyze in the same physical deployment topology as the target production deployment. For example, you might modify the configuration to deploy the components of i2 Analyze on multiple servers or with high availability.

Creating the pre-production environment

In the pre-production environment, you can deploy i2 Analyze in the same physical deployment topology as your target production deployment. The process starts with your configuration from the configuration development environment, which you modify to match your chosen deployment topology.

Ensure that you have access to the following files and directories so that you can copy them to the pre-production development environment:

- The `toolkit\configuration` directory from the configuration development environment
- Liberty user configuration, for example:
 - `user-registry.xml`
 - `server.xml`
- Any configuration that you completed in the Information Store database. For example, your merged property values definition views.
- Any certificates and certificate stores that are required

If you are planning to deploy in production with high availability, IBM recommends that you configure and deploy high availability in the pre-production environment to develop the configuration. To create a pre-production environment configured for high availability, complete the instructions in [“i2 Analyze with high availability”](#) on page 74 rather than the steps on this page.

Create your pre-production environment, and update your configuration to match the physical deployment topology of the pre-production environment.

1. Install any prerequisite software to prepare your servers for the pre-production environment.
For the pre-production environment, use the same deployment topology as your intended production environment. For more information, see [“Deployment topologies”](#) on page 54.
2. Copy the `toolkit\configuration` directory from the configuration development environment, to the `toolkit` directory at the root of the deployment toolkit on the Liberty server in the pre-production environment.
3. Update the values for any configuration settings that are specific to the environment.
 - a) If you are creating a deployment with a database that is remote from the Liberty server, follow the instructions in [“Specifying remote database storage”](#) on page 68 to update the `environment.properties` and `topology.xml` files for this deployment topology.
 - b) If you are creating a deployment with multiple Solr and ZooKeeper servers, follow the instructions in [“Specifying remote Solr and ZooKeeper servers”](#) on page 71 to update the `topology.xml` file for these deployment topologies.

The `environment.properties`, `http-server.properties`, and `topology.xml` contain host name and file path settings that you might need to update for the servers in your pre-production environment. For more information, see [Configuration files reference](#).

4. If your deployment uses a proxy server or load balancer to route requests from clients, ensure that it is configured for use with i2 Analyze. Specify the URI that clients use to connect to i2 Analyze.
For more information, see [“Deploying a proxy server for use with i2 Analyze”](#) on page 73.
5. Deploy and start i2 Analyze:
 - In a single-server topology, or with a remote database only, see [“Deploying i2 Analyze”](#) on page 85.
 - In a multiple-server topology, see [“Deploying i2 Analyze on multiple servers”](#) on page 86.

After you deploy i2 Analyze, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze.

6. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment.

This might involve copying the user registry file, or updating the server.xml file.

7. Complete any configuration changes in the Information Store database.

a) If you created any rules or schedules to delete records by rule, replicate the rules and schedules that you created in the previous environment.

b) If you created any merged property values definition views for your ingestion process, replicate the view definition that you created in the previous environment.

After you deploy i2 Analyze in the pre-production development environment, you might want to configure aspects of the deployment that are topology specific. For example, in a multiple-server environment you can secure the connections between servers by using SSL.

To configure i2 Analyze in pre-production, see [“Configuring i2 Analyze in pre-production”](#) on page 50.

Configuring i2 Analyze in pre-production

In the pre-production environment, configure the aspects of i2 Analyze that depend on the deployment topology.

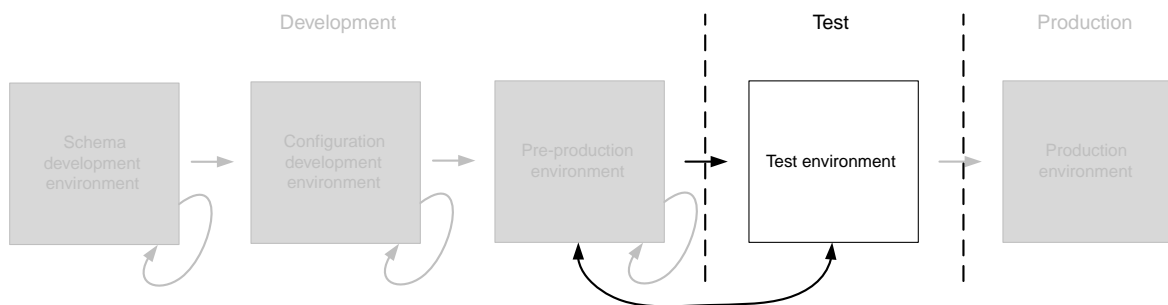
1. To configure SSL connections between the servers in your deployment, refer to [Secure Sockets Layer connections with i2 Analyze](#).
2. To develop the process for backing up and restoring your deployment of i2 Analyze, refer to [Backing up a deployment](#).
3. To understand how to administer high availability and develop your continuous operation processes, refer to [Administering high availability](#).

After you configure i2 Analyze sufficiently, you can move to the test environment. Keep your pre-production environment in place so that you can access the configuration directory in later phases of the production process, and return to it to make further configuration changes.

Next, create the [“Test environment”](#) on page 50.

Test environment

The second phase of creating a production deployment focuses on testing. This phase is important, because it enables you to identify any changes to the environment or configuration that must be completed before you deploy into production.



You use a *test environment* to test the deployment to ensure that it meets the requirements of the production deployment. The test environment should match the production environment as closely as possible. In the test environment, perform comprehensive testing of the deployment against the final requirements with a selection of the users and a sample data set.

Creating the test environment

In the test environment, you deploy i2 Analyze with the configuration that you developed previously. With i2 Analyze deployed in the test environment, you can perform comprehensive testing of the deployment to confirm that it meets your production requirements.

Ensure that you have access to the following files and directories so that you can copy them to the test environment:

- The `toolkit\configuration` directory from the pre-production environment
- Liberty user configuration, for example the `server.xml` file
- Any configuration that you completed in the Information Store database. For example, your merged property values definition views.
- Any certificates and certificate stores that are required

If you are planning to deploy in production with high availability, complete the instructions in [“i2 Analyze with high availability”](#) on page 74 rather than the steps on this page.

1. Install any prerequisite software to prepare your servers for the test environment.
For the test environment, use the same deployment topology as in your pre-production environment. For more information, see [“Deployment topologies”](#) on page 54.
2. Copy the `toolkit\configuration` directory from the pre-production environment to the `toolkit` directory at the root of the deployment toolkit on the Liberty server in the test environment.
3. Update the values for any configuration settings that are specific to the environment.
`environment.properties`, `http-server.properties`, and `topology.xml` contain settings that you might need to update. For more information, see [Configuration files reference](#).
4. If your deployment uses a proxy server or load balancer to route requests from clients, ensure that it is configured for use with i2 Analyze. Specify the URI that clients use to connect to i2 Analyze.
For more information, see [“Deploying a proxy server for use with i2 Analyze”](#) on page 73.
5. Deploy and start i2 Analyze:
 - In a single-server topology, or with a remote database only, see [“Deploying i2 Analyze”](#) on page 85.
 - In a multiple-server topology, see [“Deploying i2 Analyze on multiple servers”](#) on page 86.

After you deploy, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze.

6. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment.
This might involve copying the user registry file, or updating the `server.xml` file.
7. Complete any configuration changes in the Information Store database.
 - a) If you created any rules or schedules to delete records by rule, replicate the rules and schedules in the current environment.
 - b) If you created any merged property values definition views for your ingestion process, replicate the view definitions in the current environment.

After you deploy i2 Analyze in the test environment, you can test your deployment to ensure that it meets the requirements of your organization.

For information about testing i2 Analyze, see [“Testing your deployment”](#) on page 52.

Testing your deployment

In your test environment, complete the comprehensive testing of the system that is required to prove that the deployment is ready for production.

Create a test charter that contains a list of tests that you can use to verify that your deployment is ready for production.

In your test environment, you should use real data at a representative volume.

1. Create the test charter.

Focus your test charter in the following areas that are applicable to your deployment:

- Data ingestion into the Information Store
- Data deletion from the Information Store
- Data acquisition from external sources
- Searching and analyzing data
- Storing and retrieving charts in the Chart Store
- Performance of the system
- Backup and recovery of the system
- High availability and continuous operations after server failure

After you create a test charter for your deployment, you can start your testing.

2. Complete each test in your charter, and record if the deployment passed or failed each test.

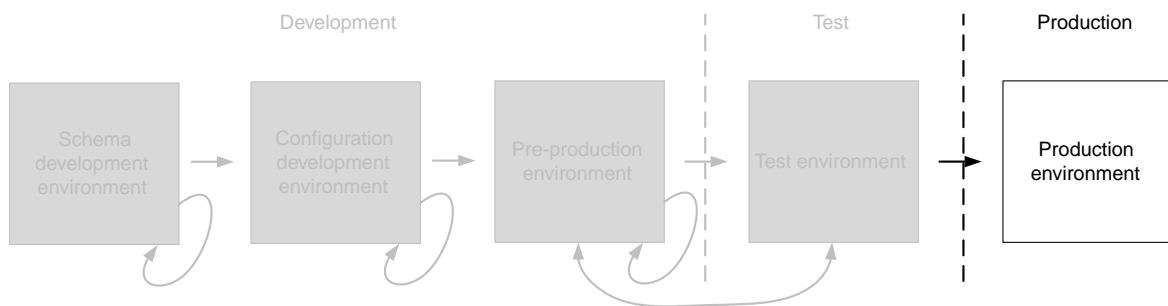
3. If any of the tests fail, return to your development or pre-production environment and change the configuration. Then, deploy the same change to the test environment and repeat the testing.

After you test i2 Analyze sufficiently, you can move to the production phase. Keep your test environment in place, so that you can access the configuration directory in later phases of the production process and continue to test any changes to your configuration.

Next, complete the instructions in [“Production environment”](#) on page 52.

Production environment

The third phase of the deployment process focuses on deploying into production. In this phase you make your deployment of i2 Analyze available to users.



You use a *production environment* to host the deployment in production. When you have a configuration of i2 Analyze that passed your test phase, you can deploy i2 Analyze with that configuration into your production environment.

In the production environment, run the tests again to confirm that the deployment is working successfully. If these tests are successful, you can make the deployment available to users. If the tests are not successful, you must return to your test or development environment to make any necessary changes. Then, complete the test phase once more.

Creating the production environment

To deploy i2 Analyze in production, you use the i2 Analyze configuration that you developed previously. Before you deploy i2 Analyze, update the configuration to reflect any environment changes.

Ensure that you have access to the following files and directories so that you can copy them to the production environment:

- The `toolkit\configuration` directory from the test environment
- Liberty user configuration, for example the `server.xml` file
- Any configuration that you completed in the Information Store database. For example, your merged property values definition views.
- Any certificates and certificate stores that are required

If you are planning to deploy with high availability, complete the instructions in [“i2 Analyze with high availability”](#) on page 74 rather than the steps on this page.

1. Install any prerequisite software to prepare your servers for the production environment.
For the production environment, use the same deployment topology as in your test environment. For more information, see [“Deployment topologies”](#) on page 54.
2. Copy the `toolkit\configuration` directory from the test environment to the `toolkit` directory at the root of the deployment toolkit on the Liberty server in the production environment.
3. Update the values for any configuration settings that are specific to the environment.
`environment.properties`, `http-server.properties`, and `topology.xml` contain settings that you might need to update. For more information, see [Configuration files reference](#).
4. If your deployment uses a proxy server or load balancer to route requests from clients, ensure that it is configured for use with i2 Analyze. Specify the URI that clients use to connect to i2 Analyze. For more information, see [“Deploying a proxy server for use with i2 Analyze”](#) on page 73.
5. Deploy and start i2 Analyze:
 - In a single-server topology, or with a remote database only, see [“Deploying i2 Analyze”](#) on page 85.
 - In a multiple-server topology, see [“Deploying i2 Analyze on multiple servers”](#) on page 86.

After you deploy i2 Analyze, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze.

6. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment.
This might involve copying the user registry file, or updating the `server.xml` file.
7. Complete any configuration changes in the Information Store database.
 - a) If you created any rules or schedules to delete records by rule, replicate the rules and schedules in the current environment.

- b) If you created any merged property values definition views for your ingestion process, replicate the view definitions in the current environment.

Complete testing of the deployment to ensure that it is working in the production environment before you make i2 Analyze available to users.

Deployment resources

The deployment resources section contains information that is referenced elsewhere in the deployment section. The information is used in a many places throughout the deployment process.

Deployment topologies

You can use the deployment toolkit to deploy i2 Analyze in a number of different physical topologies. Throughout the process of creating a production deployment, you might use several of them depending on your purpose at the time.

The following diagrams show the servers that are used in each deployment topology, the prerequisites that are required on each server, and the components of i2 Analyze that are deployed.

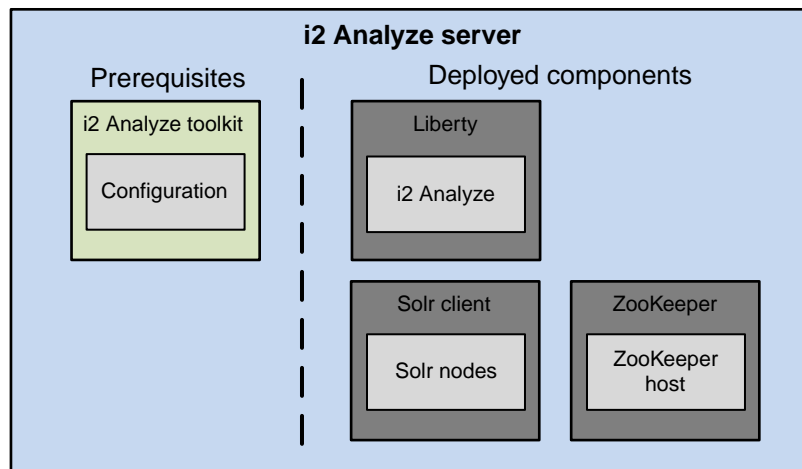
You can deploy i2 Analyze in the following physical deployment topologies:

- [“i2 Connect gateway only \(single server\)” on page 54](#)
- [“Single server” on page 55](#)
- [“Multiple servers” on page 55](#)

i2 Connect gateway only (single server)

In the i2 Connect gateway only deployment, all of the components of i2 Analyze are on the same server. No database management system is required.

i2 Connect only topology



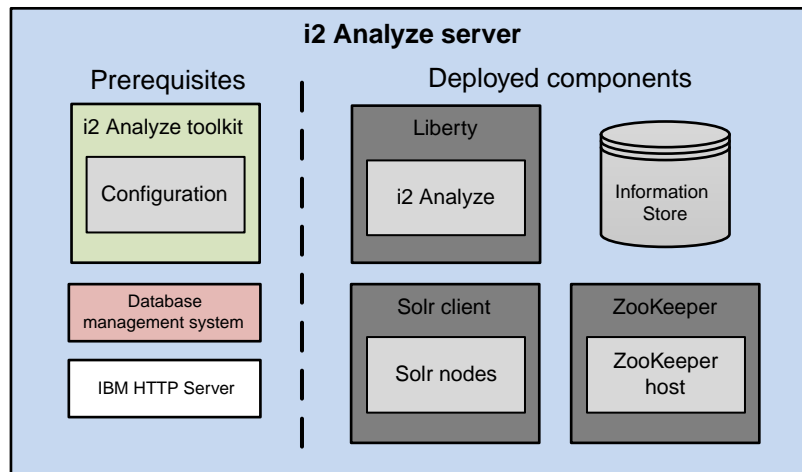
On the i2 Analyze server, install the prerequisites by following these instructions:

- [Installing i2 Analyze](#)

Single server

In the single-server topology, all of the components of i2 Analyze are on the same server.

Single-server topology



On the i2 Analyze server, install the prerequisites by following these instructions:

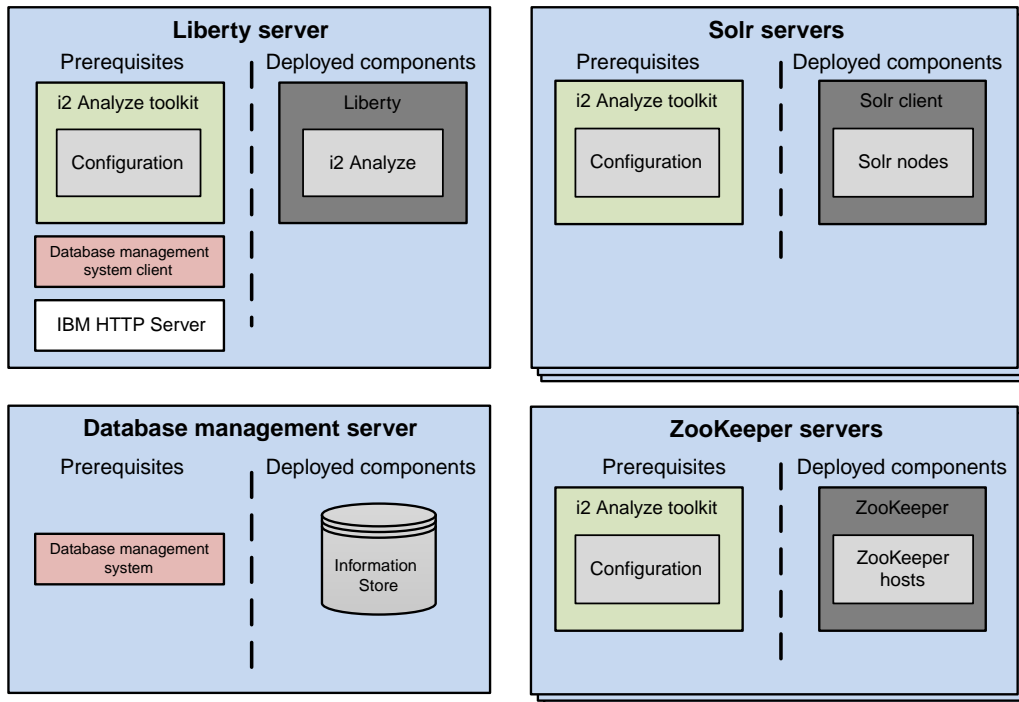
- [Installing i2 Analyze](#)
- [Software prerequisites](#)

The HTTP server is an optional part of a single-server topology.

Multiple servers

In the multiple-server topology, each component of i2 Analyze is on its own server.

Multiple-server topology



In the diagram, Liberty, Solr, ZooKeeper, and the Information Store (or the Chart Store) are each deployed on their own server.

You can also have a multiple-server deployment topology in which any of number of the components are located on the same server.

On the Liberty server, install i2 Analyze, the database management system client, and optionally the HTTP server:

- [Installing i2 Analyze](#)
- [Db2 client](#) or [SQL Server client](#)
- [HTTP Server](#)

On your database management server, install your database management system:

- [Installing Db2](#) or [Installing SQL Server](#)

On each Solr and ZooKeeper server, install i2 Analyze:

- [Installing i2 Analyze](#)

The i2 Analyze deployment toolkit

The deployment toolkit contains scripts and components that you need to deploy and maintain i2 Analyze. Configuring components within the toolkit allows you to deploy consistently to the same locations with the same settings.

Directories

The deployment toolkit contains files in several directories. On most occasions, you need to interact with only three of the directories:

- The `examples` directory includes the base configurations that you can use to create example deployments, example data, and an example i2 Connect connector.
- The `configuration` directory contains files that you must update with information specific to your deployment.

Note: When the deployment toolkit is first installed, this directory does not exist. At the start of deployment, you use one of the base configurations to create this directory.

- The `scripts` directory contains the setup script that you use to deploy and configure i2 Analyze.

The setup script

The setup script completes tasks that apply configuration and other changes to deployments of i2 Analyze. For a list of available tasks and other information, refer to [“Deployment toolkit tasks” on page 59](#).

Alternatively, on the i2 Analyze server, open a command prompt and navigate to `toolkit\scripts` and run one of the following commands:

setup -h

The `-h` argument displays the usage, common tasks, and examples of use for the setup script.

setup -a

The `-a` argument displays the same content as when you use `-h`, and a list of additional tasks.

Note: On Linux, whenever you run the setup script you must prefix its name with `./`. For example, `./setup -t deploy`.

The base configurations

In the i2 Analyze deployment toolkit, several base configurations are available. To reduce the amount of configuration that you need to complete, you can start from the configuration that is closest to the requirements of your final deployment pattern.

The i2 Analyze deployment toolkit includes the following base configurations:

chart-storage

The `chart-storage` base configuration contains settings for a deployment of i2 Analyze that includes:

- A Chart Store only

Users can connect to the deployment with the following client:

- Analyst's Notebook Premium

information-store-opal

The `information-store-opal` base configuration contains settings for a deployment of i2 Analyze that includes:

- An Information Store (and a Chart Store)

Users can connect to the deployment with the following clients:

- Analyst's Notebook Premium
- i2 Notebook web client
- i2 Investigate web client

daod-opal

The `daod-opal` base configuration contains settings for a deployment of i2 Analyze that includes:

- The i2 Connect gateway only, which can provide access to other data stores through connectors that you create

Users can connect to the deployment with the following client:

- Analyst's Notebook Premium

chart-storage-daod

The `chart-storage-daod` base configuration contains settings for a deployment of i2 Analyze that includes:

- A Chart Store
- The i2 Connect gateway, which can provide access to other data stores through connectors that you create

Users can connect to the deployment with the following clients:

- Analyst's Notebook Premium
- i2 Notebook web client

information-store-daod-opal

The `information-store-daod-opal` base configuration contains settings for a deployment of i2 Analyze that includes:

- An Information Store (and a Chart Store)
- The i2 Connect gateway, which can provide access to other data stores through connectors that you create

Users can connect to the deployment with the following clients:

- Analyst's Notebook Premium
- i2 Notebook web client
- i2 Investigate web client

It is possible to extend your deployment after you select and deploy a base configuration by adding the appropriate components and configuration files.

For more information about the components that are included in a deployment of i2 Analyze, see [Understanding components](#).

Example schemas

The i2 Analyze deployment toolkit includes example schema files that you can use as a starting point for your own schemas.

Example i2 Analyze schemas

The deployment toolkit includes five pairs of example i2 Analyze schemas and associated charting schemes in the `toolkit\examples\schemas` directory:

Law Enforcement

The law enforcement schema deals with criminal activity. It contains entity and link types that are designed to track connections within criminal networks.

Commercial Insurance

The commercial insurance schema deals with fraud in a commercial setting. It contains entity and link types that are designed to track financial transactions such as credit card payments and insurance claims.

Military

The military schema helps with military intelligence tracking. It contains entity and link types that target military operations.

Signals Intelligence

The signals intelligence schema focuses particularly on the cellphones and cell towers that are involved in mobile telecommunications, and on the calls that take place between them.

Chart Storage

The chart storage schema is not an example of a particular domain, but rather the starting point for any deployment of i2 Analyze that includes only the Chart Store.

Example security schema

The example security schema files specify a number of security dimensions and dimension values, and security groups for users.

After you create the `toolkit\configuration` directory by copying it from the [base configuration](#) of your choice, you can find an appropriate example security schema file in the `toolkit\configuration\examples\security-schema` directory.

Deployment toolkit tasks

The following toolkit options and tasks are available to use with the setup script in the deployment toolkit.

usage: setup [-h] [-a] [-s SERVER] [-w WAR | -id IDENTIFIERS | -hn HOST | --all] [--force] -t TASK [--option-name...]

Argument name	Argument Description
-h, --help	Shows this help message, and exits
-a, --additional	Shows additional arguments and tasks, and exits
-t, --task <task>	Specifies the task to perform
-s, --server <server>	Specifies the server profile to manage

Argument name	Argument Description
-w, --war <war>	Specifies the war name
-co, --collection <collection>	Specifies the solr collection ID
-id, --id <ids>	Specifies the ids of the components to manage
-hn, --hostname <hostname>	Restrict the task to operate only on components with the specified hostname attribute
-l, --locale <locale>	Specify the language code of the schema files to use in the example deployment
--all	Apply the operation to all applicable components
-sn, --schemaName <Schema>	Specify the name of the schema file to use in the example deployment
-e, --exampleData <Example Data>	Specify the name of the directory containing the example data to ingest
-f, --force	Suppress the warning prompt that is associated with tasks that might result in data loss
--scripts	Generates scripts
--quiet	Show QUIET and higher log messages
--warn	Show WARN and higher log messages
--lifecycle	Show LIFECYCLE and higher log messages
--info	Show INFO and higher log messages
-d, --debug	Show DEBUG and higher (that is, all log messages)
-st, --stacktrace	Print the full stacktrace if an error occurs

The following installation and deployment tasks are available:

Task name	Task Description
installLiberty	Installs Liberty from the Liberty binaries in the toolkit, to the directory specified in environment.properties.
installZookeeper	Installs ZooKeeper from the ZooKeeper binaries in the toolkit, to the directory specified in environment.properties.
installSolr	Installs Solr from the Solr binaries in the toolkit, to the directory specified in environment.properties.
deployExample	Deploys i2 Analyze with default files and settings.
ingestExampleData	Ingests entity and link record examples into the Information Store.
deploy	Creates the databases, creates the application profile, and deploys i2 Analyze.
deployLiberty	Deploys the i2 Analyze application into Liberty.
start	Starts all i2 Analyze services on the current server.

Task name	Task Description
stop	Stops all i2 Analyze services on the current server.
restart	Restarts all i2 Analyze services on the current server.
configSummary	Summarizes the configuration of the toolkit.
version	Summarizes the deployment version information.

Examples of use:

- `setup -t deployExample`
- `setup -t ingestExampleData`
- `setup -t deploy`
- `setup -t start`
- `setup -t configSummary`

The following upgrade tasks are available:

Task name	Task Description
upgrade	Upgrades the configuration files, and upgrades i2 Analyze.
upgradeConfiguration	Upgrades the configuration files for an Information Store deployment.
upgradeZookeeper	Upgrades ZooKeeper and ZooKeeper configuration files to the version required by the toolkit.
upgradeSolr	Upgrades Solr and Solr configuration files to the version required by the toolkit.
upgradeDatabases	Upgrades the Information Store database and clears the search index if required.
upgradeSolrCollections	Upgrades ZooKeeper and Solr, and creates a new collection if required.
upgradeLiberty	Upgrades Liberty to the version required by the toolkit.

Examples of use:

- `setup -t upgrade`
- `setup -t upgradeConfiguration`
- `setup -t upgradeSolr -hn "example.solr.hostname"`

The "upgradeZookeeper", "upgradeSolr", "upgradeDatabases", and "upgradeSolrCollections" tasks support an optional `-hn` argument that restricts their effect to a single host.

The following administration tasks are available:

Task name	Task Description
replayFromTimestamp	Starts Liberty in a mode that replays all events since the time specified by the 'datetime.to.replay.from' property in environment-advanced.properties.

Task name	Task Description
configureHttpServer	Sets up the reverse proxy configuration for IBM HTTP Server.
enableLibertyAdminCenter	Enable the Liberty Admin Center.
disableLibertyAdminCenter	Disable the Liberty Admin Center.
generateDefaults	Configures the environment with default property values.
ensureBasicUserRegistry	Configures the application for basic user registry authentication.
ensureExampleUserRegistry	Populates the user registry with an example user and user groups that map to the example security schema.
ensureDefaultUserProfileProvider	Configures the application with the default user profile provider.
addInformationStore	Generates a fragment for the Information Store, and updates topology.xml.
addI2Connect	Updates topology.xml for i2 Connect.
updateConnectorsConfiguration	Updates the i2 Analyze server with the connection details of the connectors defined in topology.xml. In a deployment without the Information Store, the i2 Analyze schema and charting schemes are also updated.
updateSchema	Updates i2 Analyze to conform to the schema file referenced in the configuration.
updateSecuritySchema	Updates i2 Analyze to use the security schema file referenced in the configuration.
updateLiveConfiguration	Updates the i2 Analyze server with the latest version of the files in the configuration/live directory from the deployment toolkit.
createDatabaseStorage	Creates the database storage*
createDatabases	Creates the database storage and tables*
modifyInformationStoreDatabase	Runs the informationStoreModifications.sql script on the Information Store database.
dropTables	Drops all of the tables from the database but leaves the database intact*
dropDatabases	Drops the entire database and de-allocates storage*
emptyInformationStore	Empties the Information Store of data, apart from metadata.
addInformationStoreIngestionSource	Adds or replaces information about an ingestion source to the Information Store.

Task name	Task Description
createInformationStoreStagingTable	Creates an Information Store staging table for a specific entity or link type.
ingestInformationStoreRecords	Ingests records into the Information Store.
deleteProvenance	Deletes (entity/link) provenance from the Information Store.
previewDeleteProvenance	Previews deleting (entity/link) provenance from the Information Store.
syncInformationStoreCorrelation	Synchronizes data in the Information Store after a correlation operation failed during ingestion.
enableMergedPropertyValues	Creates the database views used to define the property values of merged i2 Analyze records.
disableMergedPropertyValues	Removes the database views used to define the property values of merged i2 Analyze records.
duplicateProvenanceCheck	Checks the Information Store for duplicated origin identifiers. Any provenance that has a duplicated origin identifier is added to a staging table.
duplicateProvenanceDelete	Deletes (entity/link) provenance from the Information Store that has duplicated origin identifiers. The provenance to delete is identified in the staging tables created by the duplicateProvenanceCheck task.
deleteOrphanedDatabaseObjects	Deletes (entity/link) database objects that are not associated with an i2 Analyze record from the Information Store.
createEt1Toolkit	Creates a DataStage ETL toolkit that contains the files DataStage requires to run pipeline jobs.
generateInformationStoreIndexCreationScripts	Generates InfoStore 'create indexes' DDL scripts for the specified item type
generateInformationStoreIndexDropScripts	Generates InfoStore 'drop indexes' DDL scripts for the specified item type
clearData	Clears the search index and all the data in the database.
clearSearchIndex	Clears the search index.
clearLTPAkeys	Clears the LTPA keys.
clearInformationStoreStagingSchema	Clears all the tables in the Information Store Staging Schema.
dropInformationStoreErrorTables	Removes the _ERROR and _REJECT tables from the Information Store.
backupDatabases	Backs up the database. i2 Analyze must be stopped first.

Task name	Task Description
restoreDatabases	Restores the database from a specified timestamp. i2 Analyze must be stopped first.
backupSolr	Backs up the Solr index and ZooKeeper configuration.
restoreSolr	Restores the Solr index and ZooKeeper configuration from a specified timestamp. i2 Analyze must be stopped first.
backupConfiguration	Backs up the i2 Analyze and Liberty configuration.
restoreConfiguration	Restores the i2 Analyze and Liberty configuration from a specified timestamp. i2 Analyze must be stopped first.
validateBackups	Validates that the timestamps of the backups specified are in the correct chronological order.

The "clearData" and "clearSearchIndex" tasks support an optional -co argument that restricts their effect to a single Solr collection.

Example of use:

- `setup -t clearData -co "solr.collection.id"`

* The exact behavior of these tasks may change depending on the chosen database engine.

Tasks for DB2 only:

Task name	Task Description
catalogRemoteDB2Nodes	Adds a remote database server entry to the DB2 node directory for each remote DB2 database that is defined in topology.xml
uncatalogRemoteDB2Nodes	Removes the remote database server entry in the DB2 node directory for each remote DB2 database that is defined in topology.xml.
recatalogRemoteDB2Nodes	Removes, then re-adds the remote database entry in the DB2 node directory for each remote DB2 database that is defined in topology.xml.
listDB2NodeDirectory	Lists the contents of the DB2 node directory.
catalogDB2Databases	Adds an entry to the system database directory for each DB2 database that is defined in topology.xml. If the database is remote from the i2 Analyze server, the database is cataloged at the node specified for that database in topology.xml.
uncatalogDB2Databases	Removes the entry in the system database directory for each DB2 database that is defined in topology.xml.
recatalogDB2Databases	Removes, then re-adds the entry in the system database directory for each DB2 database that is defined in topology.xml.
listDB2SystemDatabaseDirectory	Lists the contents of the local DB2 system database directory.

Examples of use:

- `setup -t configureHttpServer`
- `setup -t replayFromTimestamp`

In addition to the start, stop and restart tasks, the following tasks are available:

Task name	Task Description
<code>startLiberty</code>	Starts Liberty.
<code>stopLiberty</code>	Stops Liberty.
<code>restartLiberty</code>	Restarts Liberty.
<code>startSolrAndZk</code>	Starts the Solr nodes and ZooKeeper hosts.
<code>stopSolrAndZk</code>	Stops the Solr nodes and ZooKeeper hosts.
<code>restartSolrAndZk</code>	Restarts the Solr nodes and ZooKeeper hosts.
<code>startSolrNodes</code>	Starts the Solr nodes.
<code>stopSolrNodes</code>	Stops the Solr nodes.
<code>restartSolrNodes</code>	Restarts the Solr nodes.
<code>startZkHosts</code>	Starts ZooKeeper hosts.
<code>stopZkHosts</code>	Stops ZooKeeper hosts.
<code>restartZkHosts</code>	Restarts ZooKeeper hosts.

Examples of use:

- `setup -t startSolrAndZk`
- `setup -t stopZkHosts -id "1,3"`
- `setup -t restartSolrNodes -id node1`
- `setup -t startSolrNodes -hn "example.solr.hostname"`

The "SolrNodes" and "ZkHosts" tasks support an optional `-id` argument.

The comma-separated list of identifiers that you specify restricts the task to the nodes and hosts with matching identifiers in the topology.

The "SolrNodes" and "ZkHosts" tasks support an optional `-hn` argument that restricts their effect to a single host.

The following Solr and ZooKeeper tasks are available:

Task name	Task Description
<code>createSolrNodes</code>	Creates the Solr nodes that are defined in topology.xml. If the nodes already exist, their configuration is updated.
<code>createZkHosts</code>	Creates the ZooKeeper hosts that are defined in topology.xml. If the hosts already exist, their configuration is updated.

Task name	Task Description
getZkStatus	Reports the status of the ZooKeeper hosts that are defined in topology.xml
createAndUploadSolrConfig	Creates and uploads the Solr configuration to the ZooKeeper hosts.
createSolrCollections	Creates the Solr collections that are defined in topology.xml.
deleteSolrCollections	Deletes the Solr collections that are defined in topology.xml.

Examples of use:

- `setup -t createSolrNodes -hn "example.solr.hostname"`
- `setup -t createSolrCollections -co "solr.collection.id"`

These tasks support an optional `-hn` argument that restricts their effect to a single host.

The "createAndUploadSolrConfig", "createSolrCollections", and "deleteSolrCollections" tasks support an optional `-co` argument that restricts their effect to a single Solr collection.

The following tasks manage main indexes and match indexes. ZooKeeper and the application server must be running for these commands to succeed:

Task name	Task Description
switchStandbyMatchIndexToLive	Sets the standby match index to live if it is READY, and sets the previous live match index to standby.
clearStandbyMatchIndex	Deletes the contents of the standby match index and sets its state to DISABLED.
updateMatchRules	Uploads the system match rules and applies them to the standby match index, which starts BUILDING again.
switchStandbyMainIndexToLive	Sets the standby main index to live if it is READY, and sets the previous live main index to standby.
clearStandbyMainIndex	Deletes the contents of the standby main index and sets its state to DISABLED.
rebuildStandbyMainIndex	Instructs the standby main index to start BUILDING again, if it exists.
switchStandbyChartIndexToLive	Sets the standby chart index to live if it is READY, and sets the previous live chart index to standby.
clearStandbyChartIndex	Deletes the contents of the standby chart index and sets its state to DISABLED.
rebuildStandbyChartIndex	Instructs the standby chart index to start BUILDING again, if it exists.
pauseIndexPopulation	Pauses index population for a specified index or all indexes. Population is paused until the Liberty server restarts.
resumeIndexPopulation	Resumes index population for a specified index or all indexes where population is currently paused.

i2 Analyze deployment settings

Some settings in the deployment toolkit require values that are specific to the environment that you are deploying in. Although these settings can sometimes use the default values, at different times during the production process you might need to specify values that meet your environment and deployment requirements.

Specifying the deployment credentials

To allow the deployment toolkit to update the database, Lightweight Third-Party Authentication (LTPA) keys, and Solr search platform components, you must provide user names and passwords. The user names and passwords that you provide allow the system to set up and administer components of i2 Analyze, and are not used to access i2 Analyze.

Database

For the database that is identified in `topology.xml`, you must specify a user name and a password in the `credentials.properties` file. The setup script uses this information to authenticate with the database.

Note: The user that you specify must have privileges to create and populate databases in the database management system.

The database credentials are stored in the following format:

```
db.infostore.user-name=user name
db.infostore.password=password
```

For example:

```
db.infostore.user-name=admin
db.infostore.password=password
```

The `db.infostore.truststore.password` credential is used only when you configure the connection between the database and Liberty to use SSL. If you are not using SSL to secure this connection, you do not need to specify a value for the `db.infostore.truststore.password` credential. For more information about configuring SSL, see [Configure Secure Sockets Layer with i2 Analyze](#).

LTPA keys

You must provide a value for the `ltpakeys.password` property. This value is used by the system to encrypt LTPA tokens.

- For a stand-alone deployment of i2 Analyze, you can specify any value as the password.
- For a deployment of i2 Analyze that uses LTPA tokens to authenticate with other systems, you must specify the same password that those systems use.

Solr search platform

The Solr search platform is used to search data in the Information Store. You must provide values for the `solr.user-name` and `solr.password` properties. Any Solr indexes are created when i2 Analyze is first deployed, and the values that you provide here become the Solr user name and password.

When you deploy i2 Analyze, the passwords in the `credentials.properties` file are encoded.

1. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.

2. Specify the database, LTPA, and Solr credentials for your deployment:

- the database user names and passwords
- the LTPA keys password
- the Solr user name and password

3. Save and close the `credentials.properties` file.

Specifying the JDBC driver

The application server requires a JDBC driver to enable communication with the database. The JDBC driver that you provide to the deployment toolkit depends on the database management system to use with the deployment.

1. Locate the JDBC driver for your database management system.

- If you are using Db2, locate the `IBM\SQLLIB\java\db2jcc4.jar` file.
- If you are using SQL Server, download the Microsoft JDBC Driver 7.4 for SQL Server archive from <https://www.microsoft.com/en-us/download/details.aspx?id=58505>. Extract the contents of the download, and locate the `sqljdbc_7.4\enu\mssql-jdbc-7.4.1.jre11.jar` file.

2. Copy the relevant driver file for your database management system to the `toolkit\configuration\environment\common\jdbc-drivers` directory.

Specifying remote database storage

You can deploy i2 Analyze with storage that is remote from the i2 Analyze server. You can choose to configure either IBM Db2 or SQL Server database storage.

Configuring remote IBM Db2 database storage

You can deploy i2 Analyze with Db2 database storage that is remote from the i2 Analyze server. When i2 Analyze is configured to deploy with a remote instance of Db2, the database is created and updated remotely without the i2 Analyze deployment toolkit present on the server that hosts the Db2 instance.

To deploy an i2 Analyze server that uses remote Db2 database storage, you must install Db2 on your database server, and Db2 or IBM Data Server Client on the application server. Both instances of Db2 must be installed according to the specifications defined in the i2 Analyze software prerequisites. For more information about installing the prerequisites, see [Software prerequisites](#).

To deploy with remote storage, the i2 Analyze configuration must contain certain information about your Db2 instances. After you update the configuration, the specified databases can be created and updated by the deployment toolkit on the remote server.

1. Edit the `configuration\environment\topology.xml` file to specify your remote Db2 databases:

- a) Update the `host-name` and `port-number` attribute values of the `<database>` element to match the values of your remote Db2 instance.
- b) Add the `node-name` and `os-type` attributes to the `<database>` element of the database to be hosted remotely.

For example:

```
<database database-type="InfoStore" dialect="db2" instance-name="DB2"
  database-name="ISTORE" xa="false" id="infostore"
  host-name="hostname" port-number="50000"
  node-name="node1" os-type="WIN" />
```

The value for `node-name` is the name of the node to create in the Db2 node directory. The value of the `node-name` attribute must start with a letter, and have fewer than nine characters. For more information about naming in Db2, see [Naming conventions](#).

The value for `os-type` is the operating system of the remote Db2 server. The value of the `os-type` attribute must be one of the following values: AIX, UNIX, or WIN.

Note: The value of the `instance-name` attribute must match the instance name of the remote Db2 instance.

You can use the `db2level` command to get the name of your remote instance. For more information, see [db2level - Show Db2 service level command](#).

2. Edit the `configuration\environment\opal-server\environment.properties` file, to specify the details of your remote and local instance of Db2.
 - a) Ensure that the value of the `db.installation.dir` property is set for the local instance of Db2 or Data Server Client on the Liberty server.

If you are using a non-root installation, set the value for this property to the `sqllib` directory in the installation user's home directory. For example, `/home/db2admin/sqllib`.
 - b) Set the value of the `db.database.location.dir` property for the remote instance of Db2 on the database server.
3. Ensure that the users that are specified for your databases in the `configuration\environment\credentials.properties` file are valid for your remote instance of Db2 on the database server.
4. Catalog the remote node:

```
setup -t catalogRemoteDB2Nodes
```

You can complete the steps that are performed by the `catalogRemoteDB2Nodes` task manually. For example, if you are deploying a system that uses Transport Layer Security (TLS).

To catalog the remote nodes manually, you can run the `CATALOG TCPIP NODE` instead of using the `setup -t catalogRemoteDB2Nodes` command. For more information about the command, see [CATALOG TCPIP/TCPIP4/TCPIP6 NODE command](#).

The following table shows how the `CATALOG` command parameters map to the values in the `topology.xml` file:

CATALOG TCPIP NODE command parameters	<database> element attributes
TCPIP NODE <i>nodename</i>	node-name
REMOTE <i>hostname</i>	host-name
SERVER <i>port number</i>	port-number
REMOTE_INSTANCE <i>instance-name</i>	instance-name

When i2 Analyze is deployed, the Information Store database is created or updated on the remote database management system. A remote node is created with the name that is specified for the `node-name` attribute, and the database is cataloged against that node.

To check that the remote nodes and databases are cataloged, you can use the `listDB2NodeDirectory` and `listDB2SystemDatabaseDirectory` tasks after you deploy i2 Analyze:

- The `listDB2NodeDirectory` task lists the contents of the Db2 node directory.

- The `listDB2SystemDatabaseDirectory` task lists the contents of the local Db2 system database directory.

Configuring remote SQL Server database storage

You can deploy i2 Analyze with SQL Server database storage that is remote from the i2 Analyze server. When i2 Analyze is configured to deploy to a remote instance of SQL Server, the Information Store database can be created and updated remotely without the i2 Analyze deployment toolkit present on the server that hosts the SQL Server instance.

To deploy an i2 Analyze server that uses remote SQL Server database storage for the Information Store, you must install SQL Server on your database server, and the SQL Server client tools on your application server. SQL Server and the client tools must be installed according to the specifications defined in the i2 Analyze software prerequisites. For more information about installing the prerequisites, see [Software prerequisites](#).

To deploy with remote storage, the deployment toolkit must contain certain information about your SQL Server installation. After you update the configuration, the specified database can be created and updated on the remote server by the deployment toolkit.

1. Edit the `configuration\environment\topology.xml` file to specify the details of your remote database:
 - a) Update the `host-name`, and either the `port-number` or `instance-name`, attribute values of the `<database>` element for the Information Store to match the values for SQL Server on your database server.
If you are using SQL Server on Linux, you must use the `port-number` attribute.
 - b) Add the `os-type` attribute to the `<database>` element for the Information Store database. The value of the `os-type` is used to support the search functions for the Information Store.
For example:

```
<database database-type="InfoStore" dialect="sqlserver"
  instance-name="remote.instance.name"
  database-name="ISTORE" xa="false" id="infostore"
  host-name="remote.hostname" os-type="WIN" />
```

Where the value for `os-type` is the operating system of the remote database server. The value of the `os-type` attribute must be one of the following values: UNIX or WIN.

For more information about the `<database>` element attributes, see [“Databases” on page 286](#).

2. Edit the `configuration\environment\opal-server\environment.properties` file to specify the details of your remote and local installations of SQL Server:
 - a) Ensure that the value of the `db.installation.dir` property is set for the local installation of SQL Server or the Microsoft Command Line Utilities for SQL Server on the Liberty server.
 - b) Set the value of the `db.database.location.dir` property for the remote installation of SQL Server on the database server.
3. Ensure that the user that is specified for your database in the `configuration\environment\credentials.properties` file is valid for your remote installation of SQL Server on the database server.

When i2 Analyze is deployed, the Information Store database is created or updated on the remote database management system.

If the connection details for the remote database management system change, you can update the `topology.xml` file and redeploy the system.

Specifying remote Solr and ZooKeeper servers

In a deployment of i2 Analyze, the location of each Solr server and ZooKeeper server is specified in the `topology.xml` file. When you run the commands to deploy i2 Analyze, the Solr nodes and ZooKeeper servers are created on servers that you specify.

Add the information about your Solr and ZooKeeper servers to the `configuration\environment\topology.xml` file.

1. In the `topology.xml` file, add at least one `<solr-node>` element for each Solr server in your deployment topology.

For example:

```
<solr-nodes>
  ...
  <solr-node
    memory="2g"
    data-dir="C:/IBM/i2analyze/data/solr"
    host-name="solr_server_host_name"
    id="node2"
    port-number="8984"
  />
</solr-nodes>
```

Where `solr_server_host_name` is the hostname of a Solr server.

For more information about the possible values for each attribute, see [Solr](#).

2. Edit the `configuration\environment\topology.xml` file to specify your ZooKeeper servers.

For example:

```
<zookeeper id="zoo">
  <zkhhosts>
    <zkhost
      host-name="zookeeper_server1_host_name" id="1"
      port-number="9983" quorum-port-number="10483" leader-port-number="10983"
      data-dir="C:/IBM/i2analyze/data/zookeeper"
    />
    <zkhost
      host-name="zookeeper_server2_host_name" id="2"
      port-number="9983" quorum-port-number="10483" leader-port-number="10983"
      data-dir="C:/IBM/i2analyze/data/zookeeper"
    />
    <zkhost
      host-name="zookeeper_server3_host_name" id="3"
      port-number="9983" quorum-port-number="10483" leader-port-number="10983"
      data-dir="C:/IBM/i2analyze/data/zookeeper"
    />
  </zkhhosts>
</zookeeper>
```

Where `zookeeper_serverx_host_name` is the hostname of the ZooKeeper server.

Each `<zkhost>` element must have a unique value for the `id` attribute.

For more information about the possible values for each attribute, see [ZooKeeper](#).

Specifying the Java distribution

i2 Analyze comes pre-packaged with the Adopt OpenJDK 11 HotSpot distribution of Java™. You can change the distribution of Java that is used, and also you can specify different distributions for different components of i2 Analyze.

Any Java distribution that you specify for use with i2 Analyze and its components must be Java version 11.

Note: If you decide to use a different distribution, you assume responsibility for updating that distribution for security patches or if the version required by the i2 Analyze changes.

To change the Java distribution to use, you modify the `setup.in` script in the `configuration\environment` directory to contain the path where it is installed. If you are using the Linux® distribution of the i2 Analyze toolkit, this is the `setup.in` file. On Windows it is `setup.in.bat`.

The `setup.in` file contains the following variables:

I2A_TOOLKIT_SCRIPTS_JAVA_HOME

The path to a Java installation that is used to run the toolkit tasks.

The path can be absolute or a relative path from the root of the toolkit.

By default, it is `\tools\java` on Windows and `tools/java` on Linux.

Note: When you deploy i2 Analyze, Adopt OpenJDK 11 HotSpot is installed in the default location. If you change the path, you must ensure that an installation of Java 11 exists in the specified location.

I2A_COMPONENTS_JAVA_HOME

An absolute path to a Java installation that is used by the Solr, ZooKeeper, and Liberty components of i2 Analyze. By default, it is `C:\IBM\i2analyze\deploy\java` on Windows and `/opt/IBM/i2analyze/deploy/java` on Linux.

Note: When you deploy i2 Analyze, Adopt OpenJDK 11 HotSpot is installed in the specified location if it does not contain an installation of Java.

The following variables are commented out in the file, but can be used to specify different Java installations for the components of i2 Analyze. If the file contains these variables, their values override the value of `I2A_COMPONENTS_JAVA_HOME` for each component.

SOLR_JAVA_HOME

An absolute path to a Java installation that is used by Solr.

Apache recommends to use Adopt OpenJDK 11 HotSpot with Solr.

ZK_JAVA_HOME

An absolute path to a Java installation that is used by ZooKeeper.

Apache recommends to use Adopt OpenJDK 11 HotSpot with ZooKeeper.

LIBERTY_JAVA_HOME

An absolute path to a Java installation that is used by Liberty.

IBM recommends to use Adopt OpenJDK 11 OpenJ9 with Liberty.

ETL_TOOLKIT_JAVA_HOME

The path to a Java installation that is used by the ingestion commands in the ETL toolkit. The path can be absolute or a relative path from the root of the ETL toolkit.

Note: When you create the ETL toolkit, Adopt OpenJDK 11 HotSpot is included as part of the ETL toolkit. If you change the path, you must ensure that an installation of Java 11 exists in the specified location.

The `setup.in` is part of the configuration, and is copied to each deployment toolkit in your environment. It is recommended that you install Java in the same location on each server where a particular component is located. This means that this file remains the same on each copy of the configuration.

1. In a text editor, open the `setup.in` script file for your operating system.
2. Uncomment and update the paths to specify the Java 11 installations that you want to use.
3. Save and close the file.

After you update the file, continue with the rest of the deployment process.

Deploying a proxy server for use with i2 Analyze

When you deploy i2 Analyze, the application is bound to the local server on port 9082. Clients can use the hostname, IP address of the server, or `localhost` to connect to the deployment. To work with your existing network, or for load-balancing purposes, you can deploy a proxy server or load balancer for use with i2 Analyze.

When you deploy a proxy server or load balancer for i2 Analyze, there are two mechanisms that you can use to determine how users connect to the application:

- Populate the `X-Forwarded-Host` and `X-Forwarded-Proto` headers on requests
- Specify a single connection string for the `FrontEndURI` setting in the i2 Analyze configuration

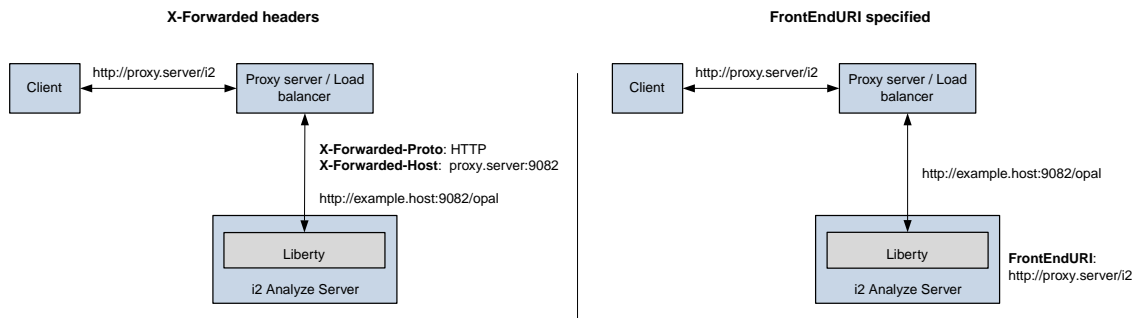
To use a proxy server or load balancer with i2 Analyze, it must meet the following requirements:

- Handle WebSocket requests and keep the WebSocket connection alive for longer than 180 seconds.
- Pass through any security configurations (for example client certificate authentication) to the i2 Analyze server. You must enable this according to the documentation for your proxy server.
- If you want to use the `X-Forwarded` headers, the proxy server or load balancer must be able to populate them. For more information about the headers, see [X-Forwarded-Host](#) and [X-Forwarded-Proto](#).
- If you are planning to deploy i2 Analyze with high availability, your load balancer must also provide server persistence. For more information, see [“Deploying a load balancer”](#) on page 83.

To allow users to connect by using the URI of the proxy server or load balancer, you can populate `X-Forwarded-Host` and `X-Forwarded-Proto` headers or specify the URI in the i2 Analyze configuration. If you configure the connection URI for your deployment, users that connect to i2 Analyze must use the URI that you specify.

If your application has the `http-server-host` attribute set to `true` in the `topology.xml`, your proxy server is not required to route requests to the port number the application is listening on.

The following image shows the URIs that you might use in this example:



1. To use the X-Forwarded headers to allow clients to connect to the deployment:

- a) Configure your proxy server or load balancer to populate the X-Forwarded-Host and X-Forwarded-Proto headers with the hostname and protocol that was used to connect to the proxy server or load balancer.

For more information about the headers, see [X-Forwarded-Host](#) and [X-Forwarded-Proto](#).

After you configure the X-Forwarded headers, you do not need to redeploy i2 Analyze. The URI that is displayed in the console will remain the same.

2. To specify a single connection URI:

- a) Using a text editor, open the `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoClientSettings.properties` file.
- b) Set the value of the `FrontEndURI` property to the URI that can be used to connect to your deployment of i2 Analyze through the proxy server.
For example, `FrontEndURI=http://proxy.server/i2`.
- c) Save and close the file.

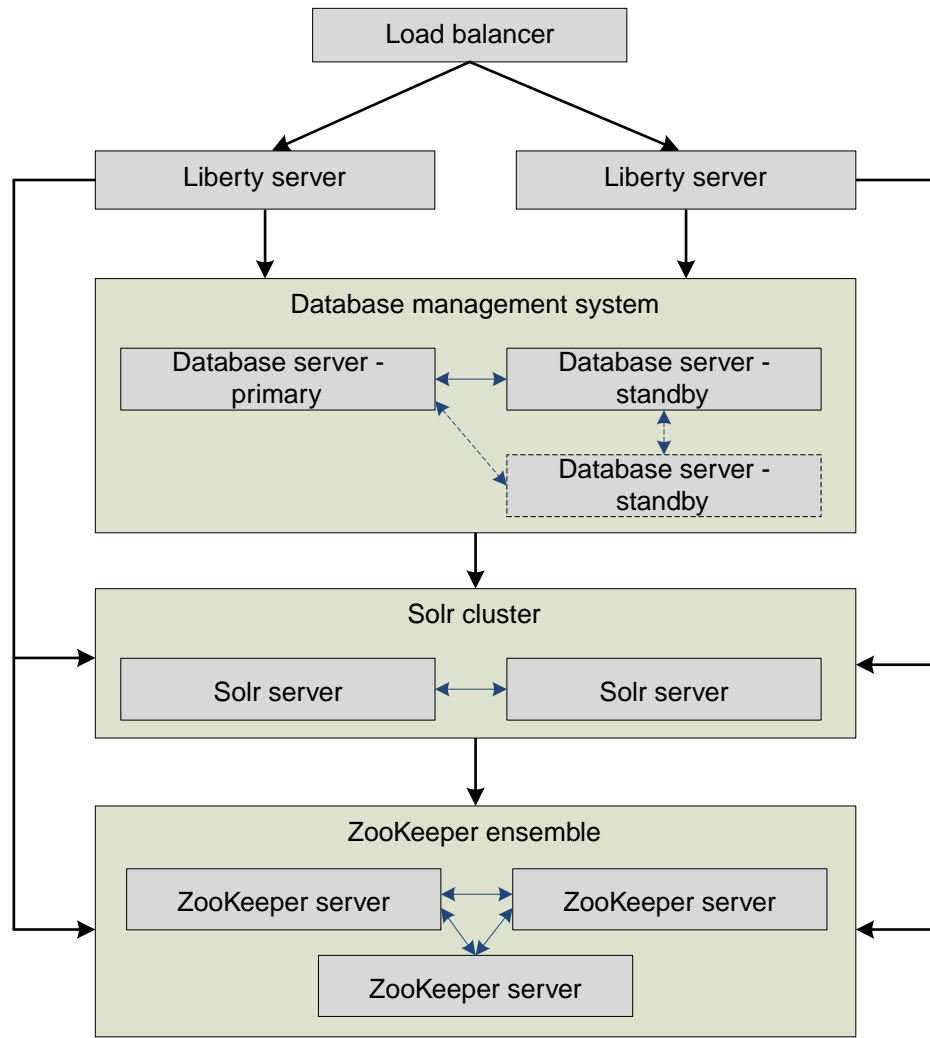
After you update the connection URI, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with your changes. After you deploy i2 Analyze and start the server, the URI that can be used to access the system is displayed in the console.

Ensure that you can access i2 Analyze by using this URI from a client workstation that uses the proxy server.

i2 Analyze with high availability

i2 Analyze supports an active/active high availability and disaster recovery deployment pattern. The only component that is not deployed in an active/active state is the database management system, which uses an active/passive pattern where one instance of the Information Store database is the primary and others are standby.

The following diagram shows the minimum number of servers for a deployment with fault tolerance for one server failure of each component:



After you read the following information that explains how each component is used in a deployment with high availability, follow the instructions to deploy i2 Analyze in this pattern. For more information about how to deploy i2 Analyze, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Load balancer

A load balancer is required to route requests from clients to the Liberty servers that host the i2 Analyze application. Additionally, the load balancer is used to monitor the status of the Liberty servers in a deployment. The load balancer must route requests only to servers that report their status as "live".

After you deploy i2 Analyze with a load balancer, you can make requests to i2 Analyze through the load balancer only.

You can also use the load balancer to distribute requests from clients across the servers in your deployment. The load balancer must provide server persistence for users.

Liberty

To provide high availability of Liberty, and the i2 Analyze application, you can deploy i2 Analyze on multiple Liberty servers. Each Liberty server can process requests from clients to connect to i2 Analyze.

In a deployment with multiple Liberty servers, one is elected the *leader*. The leader can process requests that require exclusive access to the database. The actions that are required to be completed on the leader Liberty are described in [Liberty leadership configuration](#).

At a minimum, two Liberty servers are required.

Database management system

To provide high availability of the Information Store database, use the functions provided by your database management system to replicate the primary database to at least one standby instance. i2 Analyze connects to the primary instance at one time, with the contents of the Information Store database replicated to the standby instances.

If the primary instance fails, one of the standby instances becomes the primary. When a standby becomes the primary, it must be configured to be read and writable. This means that i2 Analyze can continue to function when the initial primary server fails.

- For more information about high availability in Db2, see [High availability](#).
- For more information about high availability in SQL Server:
 - For Enterprise edition, see [What is an Always On availability group?](#)
 - For Standard edition, see [Basic availability groups](#)
 - If you are deploying with SQL Server on Linux, see [Always On availability groups on Linux](#)

For Db2, at a minimum one primary server and one standby server is required.

For SQL Server, at a minimum one primary server and two standby servers are required. If you are using basic availability groups, this is one standby server and one failover quorum witness.

Solr

i2 Analyze uses SolrCloud to deploy Solr with fault tolerance and high availability capabilities. To provide fault tolerance and high availability of the Solr cluster, deploy the cluster across at least two Solr nodes with each node on a separate server.

At a minimum, two Solr servers are required.

For more information about SolrCloud, see [How SolrCloud Works](#).

ZooKeeper

To deploy ZooKeeper for high availability, deploy multiple ZooKeeper servers in a cluster known as an *ensemble*. For a highly available solution, Apache ZooKeeper recommend that you use an odd number of ZooKeeper servers in your ensemble. The ZooKeeper ensemble continues to function while there are more than 50% of the members online. This means, that if you have three ZooKeeper servers, you can continue operations when a single ZooKeeper server fails.

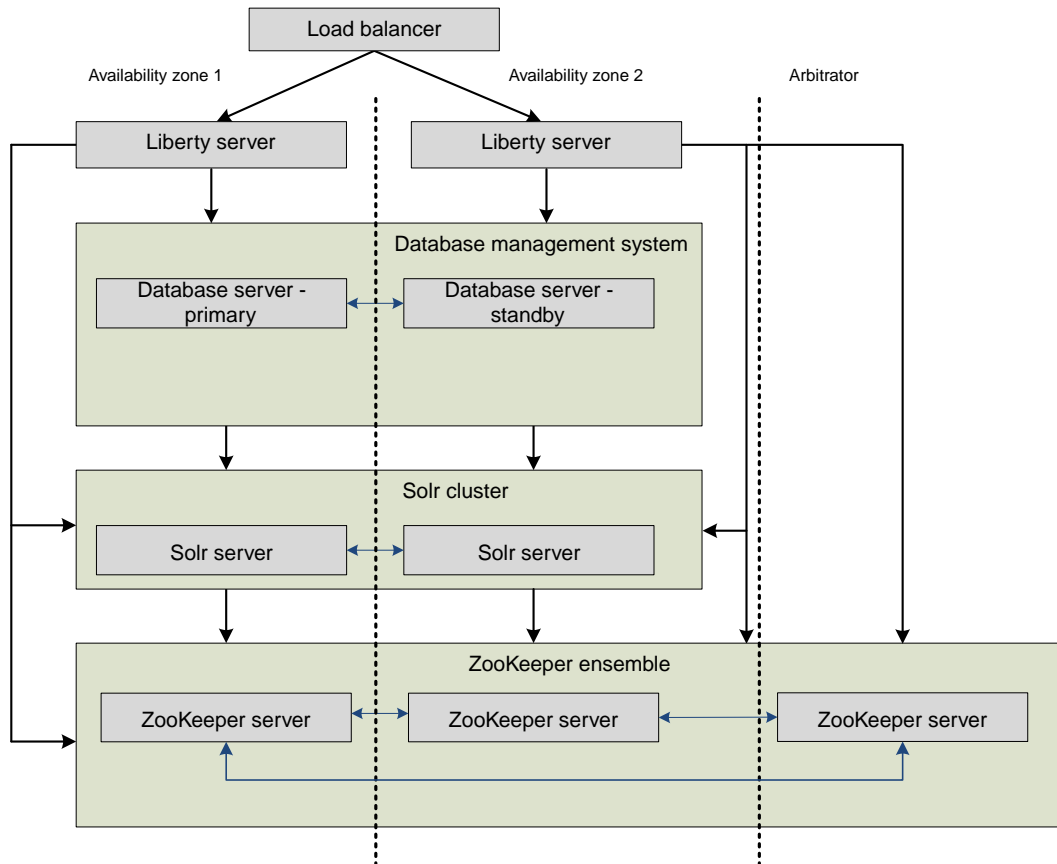
At a minimum, three ZooKeeper servers are required.

For more information about a multiple server ZooKeeper setup, see [Clustered \(Multi-Server\) Setup](#).

Deploy for disaster recovery

The deployment topology to provide availability in the event of an availability zone failure is similar to the high availability topology. An active/active pattern is still used for Liberty, Solr, and ZooKeeper, however the servers that you deploy are in separate availability zones.

The components behave in the same way as in the high availability deployment topology, except they are deployed in availability zones. You can think of an availability zone as a data center. By deploying across a number of availability zones, the deployment continues to function if one of the zones fails.



Note: For Microsoft SQL Server, you must have a third database server to act as a witness. To continue to provide availability when one of the zones fails, this server must be located outside of the two availability zones. For more information, see [Three synchronous replicas](#).

- Because ZooKeeper requires more than 50% of the servers in an ensemble to be available, one server must be located outside of the availability zones. The ZooKeeper server does not require a significant amount of resources so it can be located on any server that can communicate with the availability zones.
- You can deploy multiple Liberty and Solr servers to provide more availability at a server level within each zone.
- For the components that use an active/active pattern, servers in each availability zone are used to process requests. To maintain the performance of the system, ensure that the network that is used between availability zones is stable and provides enough bandwidth for system operation.

Deploying i2 Analyze with high availability

When you deploy i2 Analyze with high availability, there are a number of steps that you must complete.

To configure i2 Analyze for high availability, there are a number of extra configuration and deployment steps that you must complete in addition to a normal multiple server deployment. When you are deploying i2 Analyze into production with a high availability configuration, complete the following steps when you are in your pre-production, test, and production environments. After you create the high availability configuration, you can copy it between environments as usual.

1. Install any prerequisite software to prepare your servers for the pre-production environment.
 - For a deployment with high availability, use the multiple servers deployment topology with at least the minimum number of servers required for high availability of each component. For more information, see [“Deployment topologies” on page 54](#).
 - To deploy with high availability, you must configure your database management system after you install it:
 - [“Preparing Db2 for HADR” on page 79](#)
 - [“Preparing SQL Server for HADR” on page 80](#)
2. Copy the `toolkit\configuration` directory from the configuration development environment, to the `toolkit` directory at the root of the deployment toolkit on one of the Liberty servers.
3. Update the configuration to specify your remote database, follow the instructions in [“Specifying remote database storage” on page 68](#) to update the `environment.properties` and `topology.xml` files.
 - a) If you are using Db2, you can specify any alternative database location information in the `topology.xml`.
 - If you are using an automated cluster-controller such as IBM Tivoli System Automation for Multiplatforms (TSAMP), then specify the hostname and port number of the cluster.
 - If you are not using an automated cluster-controller, in the `<database>` element for your Information Store, provide the hostname and port number of any standby databases in the `alternate-hosts` attribute.

The host and port number must be separated by a colon (:), and each database server must be separated by a (,).

For example, `alternate-hosts="hostname:port_number,hostname:port_number"`.
 - b) If you are using SQL Server, specify the hostname and port number of the availability group listener.
4. Update the configuration to configure each component for high availability:
 - a) [“Configuring ZooKeeper for HADR” on page 81](#)
 - b) [“Configuring Solr for HADR” on page 82](#)
 - c) [“Deploying a load balancer” on page 83](#)

The `environment.properties`, `http-server.properties`, and `topology.xml` contain hostname and file path settings that you might need to update for the servers in your pre-production environment. For more information, see [Configuration files reference](#).
5. Deploy and start i2 Analyze with one Liberty server by following the instructions in [“Deploying i2 Analyze on multiple servers” on page 86](#).
6. After you deploy with one Liberty server, update the configuration on any other Liberty servers.

- a) Copy the configuration directory from the deployed Liberty server to the toolkit directory on any other Liberty servers in your environment.
- b) On each Liberty server, in the topology.xml file update the host-name and port-number attributes of the <application> element.

7. Deploy and start the i2 Analyze application on each of the other Liberty servers:

```
setup -t installLiberty
setup -t deployLiberty
setup -t startLiberty
```

After you deploy i2 Analyze, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze on each Liberty server.

The security configuration must be the same on each server.

8. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment.

This might involve copying the user registry file, or updating the server.xml file.

9. After you deploy i2 Analyze, configure your database management system to replicate the Information Store database to your standby servers.

- a) If you are using Db2, configure high availability for the Information Store database and replicate it to any standby database instances.

For more information, see [“Replicate the Information Store in Db2”](#) on page 84.

- b) If you are using SQL Server, add the Information Store to your availability group.

For more information, see [“Replicate the Information Store in SQL Server”](#) on page 85.

10. Complete any configuration changes in the Information Store database on the primary server.

- a) If you created any rules or schedules to delete records by rule, replicate the rules and schedules that you created in the previous environment.

On SQL Server, you must update the automated job creation schedule on every database instance. For more information see, [Changing the automated job creation schedule](#).

- b) If you created any merged property values definition views for your ingestion process, replicate the view definition that you created in the previous environment.

After you deploy i2 Analyze with high availability, return to perform the rest of the instructions for creating a deployment in your current environment:

- [“Creating the pre-production environment”](#) on page 49
- [“Creating the test environment”](#) on page 51
- [“Creating the production environment”](#) on page 53

Preparing Db2 for HADR

When you deploy i2 Analyze with multiple database servers, there must be a mechanism in place to route requests to the new primary server if the current primary server fails.

In Db2, you configure the Information Store database for HADR after you create it. However, there are some decisions that you must make before you create the i2 Analyze configuration.

When you configure i2 Analyze, you must specify the location of the remote database servers in your system. When you configure Db2 for HADR, you can use an automated cluster-controller such as IBM Tivoli System Automation for Multiplatforms (TSAMP) to ensure that clients can connect to the new primary database instance.

If you do not choose to use an automated cluster-controller, you can specify the connection properties for all of your database servers in the i2 Analyze configuration. When you provide the details in the configuration, i2 Analyze uses client-side Automatic Client Rerouting (ACR) to route requests to the new primary server.

Before you start to configure i2 Analyze, choose an approach to ensure that clients can connect to any standby database instances.

- For more information about automatic cluster-controllers, see [High availability through clustering](#).

Deploy i2 Analyze and the Information Store database. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Preparing SQL Server for HADR

To deploy i2 Analyze with HADR for the Information Store database, SQL Server must be configured to be part of an availability group.

Ensure that SQL Server is installed correctly for Always On availability groups. For more information, see [Installing Microsoft SQL Server for i2 Analyze](#).

If you are using Microsoft SQL Server Standard edition, you can use basic availability groups only. For more information about basic availability groups, see [Always On Basic Availability Groups](#).

To configure SQL Server for high availability, you add your database servers to a fail over cluster and then add any database instances to an Always On availability group.

On Windows, use Windows Server failover clustering to create a cluster of servers to host your SQL Server instances. For more information, see [Failover Clustering in Windows Server](#).

On Linux, use Pacemaker to create the server failover cluster.

If you are deploying SQL Server on Windows, complete the following steps.

1. Install the Windows Failover Clustering feature on each of the servers that will be in your SQL Server Always On Availability Group and create a failover cluster.

For more information, see [Create a failover cluster](#).

For i2 Analyze, the failover cluster must use a static IP address.

2. Create and configure an Always On availability group.

For more information, see [Reference for the creation and configuration of Always On availability groups](#)

- a) Enable Always On availability groups on each database server.

For more information, see [Enable the Always On availability group feature for a SQL Server instance](#).

- b) Create an Availability Group on the primary SQL Server instance.

For more information, see [Use the New Availability Group Dialog Box \(SQL Server Management Studio\)](#).

- Set **Required synchronized secondaries to commit** to 1.

Set it to 0 if you are using Basic availability groups.

- Select **Database level health detection**.

- Do not specify any **Availability Databases** now because the Information Store database does not exist.

- c) Create an availability group listener for the availability group.

For more information, see [Configure a listener for an Always On availability group](#).

Use a static IP address for the availability group listener. When you configure i2 Analyze to connect to a remote Information Store database, you specify the hostname and port number of the Availability Group Listener.

If you are deploying SQL Server on Linux, complete the following steps.

3. Deploy a Pacemaker cluster.

For more information, see [Deploy a Pacemaker cluster for SQL Server on Linux](#).

4. Create and configure an Always on availability group and the Pacemaker cluster.

For more information, see [Create and configure an availability group for SQL Server on Linux](#).

- Set **cluster type** to *EXTERNAL*
- In **Select Databases**, do not select any databases because the Information Store database does not exist.
- In **Specify Replicas**, create an Availability Group Listener. When you create the availability group listener, use a static IP address. When you configure i2 Analyze to connect to a remote Information Store database, you specify the hostname and port number of the Availability Group Listener.
- When you create the IP address resource for the availability group, use the same IP address as the Availability Group Listener.

Deploy i2 Analyze and the Information Store database. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Configuring ZooKeeper for HADR

The i2 Analyze configuration specifies the structure of your ZooKeeper ensemble. To provide high availability, configure a ZooKeeper ensemble with an odd number of servers.

In the i2 Analyze configuration, provide the details about the ZooKeeper servers in your environment.

Edit the configuration\environment\topology.xml file to specify your ZooKeeper servers.

For example:

```
<zookeeper id="zoo">
  <zkhhosts>
    <zkhhost
      host-name="zookeeper_server1_host_name" id="1"
      port-number="9983" quorum-port-number="10483" leader-port-number="10983"
      data-dir="C:/IBM/i2analyze/data/zookeeper"
    />
    <zkhhost
      host-name="zookeeper_server2_host_name" id="2"
      port-number="9983" quorum-port-number="10483" leader-port-number="10983"
      data-dir="C:/IBM/i2analyze/data/zookeeper"
    />
    <zkhhost
      host-name="zookeeper_server3_host_name" id="3"
      port-number="9983" quorum-port-number="10483" leader-port-number="10983"
      data-dir="C:/IBM/i2analyze/data/zookeeper"
    />
  </zkhhosts>
</zookeeper>
```

Where *zookeeper_serverx_host_name* is the hostname of the ZooKeeper server.

Each <zookeeper> element must have a unique value for the id attribute.

For more information about the possible values for each attribute, see [ZooKeeper](#).

Continue configuring the i2 Analyze configuration. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Configuring Solr for HADR

The i2 Analyze configuration specifies the structure of your Solr cluster. To provide high availability, configure a Solr cluster over at least two Solr servers.

In the i2 Analyze configuration, provide the details about the Solr servers in your environment. To provide high availability of the content of the Solr indexes, specify the number of replicas, the minimum replication factor, and the location of the replicas.

For more information about SolrCloud, see [How SolrCloud Works](#).

1. Specifying the Solr nodes.

To deploy Solr for high availability, you must have at least two Solr nodes on separate hosts.

a) Add the Solr nodes to your `topology.xml` file.

For example:

```
<solr-nodes>
  <solr-node
    memory="2g"
    data-dir="C:/IBM/i2analyze/data/solr"
    host-name="solr_server1_host_name"
    id="node1"
    port-number="8983"
  />
  <solr-node
    memory="2g"
    data-dir="C:/IBM/i2analyze/data/solr"
    host-name="solr_server2_host_name"
    id="node2"
    port-number="8983"
  />
</solr-nodes>
```

Where `solr_serverx_host_name` is the hostname of a Solr server.

For more information about the possible values for each attribute, see [Solr](#).

2. Configuring Solr replicas.

In Solr, the data is stored as *documents* in *shards*. Every shard consists of at least one *replica*. For a highly available solution, you must have more than one replica of each shard and these replicas must be distributed across the servers that host the Solr nodes.

a) Specifying the *replication factor* in your `topology.xml`.

The replication factor is the number of replicas to be created for each shard. For high availability, this must be 2 or more. You specify the replication factor in the `num-replicas` attribute of the `<solr-collection>` element.

b) Specifying the *minimum replication factor* in your `topology.xml`.

The minimum replication factor defines when data is successfully replicated in Solr. If you have three replicas for a shard and a minimum replication factor of 2, a write operation is deemed

successful if the data is written to at least two replicas. You specify the minimum replication factor in the `min-replication-factor` attribute of the `<solr-collection>` element.

The following extract from a `topology.xml` file shows an example of the `num-replicas` and `min-replication-factor` attributes:

```
<solr-collections>
  <solr-collection
    num-replicas="2"
    min-replication-factor="1"
    id="main_index"
    type="main"
    max-shards-per-node="4"
    num-shards="1"
  />
  ...
</solr-collections>
```

c) Specify the Solr autoscaling policies.

To ensure that your system can still operate if a Solr server or node fails, you must ensure that there is at least one replica of each shard still available. To do this, you can create a replica placement policy to specify that each replica of a shard must be placed on a distinct host.

- Create the `configuration\environment\opal-server\solr.autoscaling.policy.json` file.
- In the file, provide your Solr autoscaling policies. For more information about the policies, see [Autoscaling Policy and Preferences](#).
- To create a cluster policy that creates each replica of a shard on a distinct host, add the following lines to the file:

```
{
  "set-cluster-policy": [
    {
      "replica": "<2",
      "shard": "#EACH",
      "host": "#EACH"
    }
  ]
}
```

- The cluster policy is set when you create the Solr cluster as part of the deployment steps.

Continue configuring the i2 Analyze configuration. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Deploying a load balancer

In a deployment with HADR, there are multiple Liberty servers that host the i2 Analyze application. To enable analysts to connect to the deployment by using a single URI, you must use a load balancer to route requests to each Liberty server in your deployment.

For a load balancer to work successfully with i2 Analyze, it must be configured to:

- Handle WebSocket requests.
- Route requests to the i2 Analyze application on active Liberty servers only.

- Use a cookie issued by the load balancer to achieve server persistence for a user. The server persistence should last as long as possible or until the server is not active.

If the connection from the clients is secured by using SSL, you might need to implement SSL termination in the load balancer to achieve server persistence.

You can also configure your load balancer to balance requests across the Liberty servers in your deployment. The mechanism that you choose must allow for server persistence to be maintained.

For more information about load balancers and application layer server persistence, see [Load Balancing, Affinity, Persistence, Sticky Sessions: What You Need to Know](#).

1. Install your load balancer on a separate server from any other component in the deployment.
2. Configure your load balancer to identify the active Liberty servers by using the `api/v1/health/live` REST endpoint.

The `live` endpoint returns 200 when the server is live and processing requests. For more information about the endpoint, see [The health/live endpoint](#).

Call the `live` endpoint on each Liberty server in your deployment. The endpoint returns the status of a single Liberty server, and not the others in the deployment.

3. Configure your load balancer to use cookie-based server persistence.

When a user connects, if there is no cookie from the load balancer that indicates it has already connected to one of the active Liberty servers, provide it with one. When this user connects again, the cookie it received ensures that any requests are routed to the same Liberty server.

If the server that a user has persistence with is offline, the load balancer must route the request to a live Liberty server. The user's cookie must be updated to achieve persistence with the live server that it connected to.

4. Before you can connect to i2 Analyze via the load balancer, you might need to specify the connection URI that clients can use to connect to i2 Analyze.
 - a) In the `configuration\fragments\opal-services\WEB-INF\classes\DiscoClientSettings.properties` file, set the value of the `FrontEndURI` setting to the URI that can be used to connect to your deployment of i2 Analyze.

For more information, see [Specifying the connection URI](#).

Some load balancers modify the HTTP origin header, the value that you specify for the `FrontEndURI` must match the value of the HTTP origin header after it is modified by the load balancer.

Continue configuring the i2 Analyze configuration. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Replicate the Information Store in Db2

After you create the Information Store database on the primary database server, initialize HADR and replicate the Information Store database to your standby servers.

To initialize HADR for the Information Store, follow the instructions described in [Initializing high availability disaster recovery \(HADR\)](#).

When you configure the synchronize mode, it is recommended that you use the SYNC synchronize mode. For more information about the synchronization modes, see [High availability disaster recovery \(HADR\) synchronization mode](#).

At a high level, the steps are:

- Update the Information Store database configuration for HADR
- Create a backup of the Information Store database on the primary server

- Restore the backup on the standby servers
- Update the Information Store database configuration on the standby servers
- Start HADR as standby on the standby servers
- Start HADR as primary on the primary server

Continue the deployment process. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

Replicate the Information Store in SQL Server

After you create the Information Store database on the primary database server, add the database to your availability group and replicate the Information Store database to your standby servers.

1. Before you can add the Information Store database to your availability group, you must create a full backup.

For more information, see [Create a Full Database Backup](#).

2. Add the Information Store database to your availability group.

For more information, see [Add a Database to an Always On availability group](#).

3. Add any replicas to the availability group.

For more information, see [Add a secondary replica to an Always On Availability Group](#).

When you add the replicas to the availability group, configure the replicas as follows:

- Set **Failover Mode** to **Automatic**
- Set **Availability Mode** to **Synchronous commit**
- Set **Readable Secondary** to **Yes**

If you are using Basic availability groups, set it to **No**.

Continue the deployment process. For more information, see [“Deploying i2 Analyze with high availability”](#) on page 78.

i2 Analyze deployment commands

The scripts that deploy i2 Analyze depend on the values in the i2 Analyze configuration. The scripts and commands that you run depend on the topology of the deployment that you want to create.

Deploying i2 Analyze

To deploy i2 Analyze in a single-server deployment topology, you can run a script. After i2 Analyze is successfully deployed, you can start the system.

You must have an i2 Analyze configuration that is set up for a single-server, remote database only, or i2 Connect gateway only deployment topology. For more information about creating a valid configuration, see [“Creating the pre-production environment”](#) on page 49.

Run any toolkit commands from the `toolkit\scripts` directory in the deployment toolkit.

1. Deploy i2 Analyze:

```
setup -t deploy
```

2. Start i2 Analyze:

```
setup -t start
```

If an error message is displayed, refer to [“Troubleshooting the deployment process”](#) on page 88.

After you deploy and start i2 Analyze, return to perform the rest of the instructions for creating a deployment in your current environment:

- [“Creating the pre-production environment” on page 49](#)
- [“Creating the test environment” on page 51](#)
- [“Creating the production environment” on page 53](#)
- [“Deploying i2 Analyze with high availability” on page 78](#)

Deploying i2 Analyze on multiple servers

To deploy i2 Analyze in a multiple-server deployment topology, you must run the commands to install, deploy, and start the components of i2 Analyze on each server.

You must have an i2 Analyze configuration that is set up for a multiple-server physical deployment topology or a deployment with high availability:

- For more information about creating a valid multiple-server configuration, see [“Creating the pre-production environment” on page 49](#).
- For more information about creating a valid high availability configuration, see [“Deploying i2 Analyze with high availability” on page 78](#).

To deploy i2 Analyze in a multiple server deployment topology, you must provide the configuration to each deployment toolkit. Then, you can run the commands to deploy the components of i2 Analyze on each server. It is important to pay attention to which server you must run each command on, and whether you need to specify the hostname of that server.

Run any toolkit commands from the `toolkit\scripts` directory in the deployment toolkit on the specified server in your environment.

Copying the i2 Analyze configuration

The i2 Analyze configuration is required by all servers that host components of i2 Analyze. You do not have to copy the configuration to the database server, if it contains only the Information Store database and no other components.

Copy the `toolkit\configuration` from the server where you created and populated your i2 Analyze configuration, to the `toolkit` directory of the deployment toolkit on each server in your environment.

Installing components

Install the components of i2 Analyze on the servers that you have identified.

1. On the Liberty server, run the following commands:

```
setup -t installLiberty
```

2. On each ZooKeeper server, run the following command:

```
setup -t installZookeeper
```

3. On each Solr server, run the following command:

```
setup -t installSolr
```

Deploying and starting components

Deploy and start the components of i2 Analyze on the specified servers.

1. On each ZooKeeper server, create and start any ZooKeeper hosts:

```
setup -t createZkHosts --hostname zookeeper.hostname  
setup -t startZkHosts --hostname zookeeper.hostname
```

Where *zookeeper.hostname* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the `topology.xml` file.

2. On the Liberty server, run the command to upload the Solr configuration to ZooKeeper:

```
setup -t createAndUploadSolrConfig --hostname liberty.hostname
```

Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

3. On each Solr server, create and start any Solr nodes:

```
setup -t createSolrNodes --hostname solr.hostname  
setup -t startSolrNodes --hostname solr.hostname
```

Where *solr.hostname* is the hostname of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

On the Liberty server, run the commands to deploy and start a number of the components.

4. Create the Solr collections:

```
setup -t createSolrCollections --hostname liberty.hostname
```

To test that the Solr Collection is created correctly, click **Cloud** in the Solr Web UI, or you can go to `http://solr.hostname:port-number/solr/#/~cloud`. Log in with the user name and password for Solr in the `credentials.properties` file.

A horizontal tree with the collection as the root is displayed. Here you can see the breakdown of the shards, nodes, and replicas in any collections.

5. Create the Information Store database:

```
setup -t createDatabases
```

To check that the database is created correctly, connect to the database by using a database management tool.

6. Deploy the i2 Analyze application:

```
setup -t deployLiberty
```

7. If you are using IBM HTTP Server, configure the HTTP Server:

```
setup -t configureHttpServer
```

8. Start the Liberty server:

```
setup -t startLiberty
```

After you deploy and start i2 Analyze, return to perform the rest of the instructions for creating a deployment in your current environment:

- [“Creating the pre-production environment” on page 49](#)
- [“Creating the test environment” on page 51](#)
- [“Creating the production environment” on page 53](#)
- [“Deploying i2 Analyze with high availability” on page 78](#)

Troubleshooting the deployment process

i2 Analyze provides mechanisms that assist with deployment before, while, and after you do it.

Deployment toolkit validation messages

The i2 Analyze deployment toolkit validates some of the configuration files that are required for a deployment. By reading the validation messages, you can be certain that required properties have values, that property values have the correct format, and that required files are in the correct locations.

Validation takes place when you run a number of the deployment toolkit tasks. Validation can complete in the following ways:

Success

On success, the validation process displays a brief configuration summary, and the main deployment process continues. For example:

```

===== Configuration Summary (brief) =====
+ configuration/fragments/common/WEB-INF/classes/ApolloServerSettings
Mandatory.properties:
- ChartingSchemesResource='law-enforcement-schema-charting-schemes.xml'
- SchemaResource='law-enforcement-schema.xml'
- DynamicSecuritySchemaResource='security-schema.xml'
=====

```

Warning

When there is a warning, the validation process displays a brief configuration summary and a WARNINGS section. The WARNINGS section identifies settings that might not be configured correctly, but the deployment process continues. For example:

```

===== WARNINGS (1) =====
+ configuration/fragments/common/WEB-INF/classes/ApolloServerSettings
Mandatory.properties:
- The <SchemaResource> property has not been set.
=====

```

Here, the schema is not set, and so the default law enforcement schema is used.

Error

If an error occurs, the validation process displays a longer configuration summary, and an ERRORS section. The ERRORS section identifies missing values that must be present. The deployment process stops, and you must correct the errors before you attempt to deploy again. For example:

```
===== ERRORS (1) =====  
+ configuration/environment/opal-server/environment.properties:  
- "db.database.location.dir" has not been set  
=====
```

Here, the database location directory is not set, so the database cannot be configured.

Deployment progress messages

During the deployment process, i2 Analyze displays detailed messages that provide information about the state and configuration of the system.

The output from the deployment process describes each task that the setup command performs during deployment. If a task runs successfully, then only its name appears. For example:

```
:buildApplication
```

There are two reasons why a task might not run, but deployment can still proceed.

UP-TO-DATE

The task was performed earlier, or its output is already present. For example:

```
:installJDBCDrivers UP-TO-DATE
```

SKIPPED

The task is not required for this deployment. For example:

```
:importLTPAKey SKIPPED
```

If an error occurs, deployment stops in a controlled manner. i2 Analyze displays a stack trace that contains the name of the task that failed, and information about the location of the error. For example:

```
:createDatabasesIfNecessary FAILED  
  
FAILURE: Build failed with an exception.  
  
* Where:  
Script 'C:\IBM\i2analyze\toolkit\scripts\gradle\database.gradle' line: 173  
  
* What went wrong:  
Execution failed for task ':createDatabasesIfNecessary'.
```

The messages are displayed on screen and sent to the log files that are stored in the toolkit\configuration\logs directory.

Deployment log files

i2 Analyze produces logging information about deployment tasks and the transactions that take place when the system is operational. After deployment, you can check these logs to help diagnose potential issues.

The following types of logged information are available for you to review:

Deployment logs

Each time that you run the setup command, a log file is created in the deployment toolkit. The messages in these logs describe which tasks were called, whether the tasks completed successfully, and the details of any issues that occurred.

These log files contain the same information as the console output. You can find them in the `toolkit\configuration\logs` directory.

Liberty logs

In addition to the information in the deployment logs, extra information that relates to the application server is also logged in `deploy\wlp\usr\servers\opal-server\logs`.

The directory contains the following log files that you might use to help troubleshoot any issues:

console.log

The `console.log` file contains basic server status and operation messages, which are also displayed on the console.

\messages\messages.log

The `messages.log` file contains messages from Liberty.

\messages\ffdc\ffdc.log

The `ffdc.log` file contains First Failure Data Capture (FFDC) output from Liberty components (for example the database drivers that typically include selective memory dumps of diagnostic data).

\opal-services\IBM_i2_Alert_Scheduler.log

The `IBM_i2_Alert_Scheduler.log` contains messages that record when the scheduled operations for saved Visual Queries run.

\opal-services\IBM_i2_Analysis_Repository.log

The `IBM_i2_Analysis_Repository.log` file contains messages that are intercepted by log4j from the i2 Analyze application.

\opal-services\IBM_i2_General.log

The `IBM_i2_General.log` file contains any messages that are intercepted by log4j, but not from the i2 Analyze application.

\opal-services\IBM_i2_Component_Availability.log

The `IBM_i2_Solr_Cluster_Status.log` file contains entries for changes to the state of the Solr clusters in your deployment.

\opal-services\IBM_i2_Connect_Async_Query.log

The `IBM_i2_Connect_Async_Query.log` file contains messages for failures or issues related to asynchronous queries from i2 Connect connectors.

\opal-services\IBM_i2_Update_Live_Configuration.log

The `IBM_i2_Update_Live_Configuration.log` file contains messages that are displayed in the console when you run the `updateLiveConfiguration` toolkit task or use the `admin/reload` endpoint.

`\opal-services\IBM_Solr.log`

The `IBM_Solr.log` file contains messages that are directly from Solr and ZooKeeper.

Solr logs

By default, information that relates to Solr is logged in `deploy\solr\server\logs\<node port>`.

Troubleshooting specific errors

The following sections describe how to react to some more common failures that can occur during the deployment of i2 Analyze.

The results configuration file contains an invalid property type identifier

The following message is displayed if there are any identifiers in your results configuration file that are not present in your i2 Analyze schema when you start i2 Analyze.

```
# [opal-services-is] OneOffStartupServiceInitializer encountered a problem communicating with the
database that it cannot recover from.
# [opal-services-is] Exception during initialization. The application is in an unusable state.
# [opal-services-is] Errors occurred accessing system resources from ApplicationLifecycleManager:
# [opal-services-is] The results configuration file contains an invalid property type identifier:
ADD1.
```

To resolve this issue, ensure that all of the identifiers in your results configuration file are present in your i2 Analyze schema. For more information, see [Setting up search results filtering](#).

Default security dimension values provider specifies dimension values that are not in the security schema or Default security dimension values provider specified no values for the dimensions with these IDs: '[]'.

The security dimension values that are specified in the security schema for the `<DefaultSecurityDimensionValues>` element are incorrect. For more information, see [Setting default dimension values](#).

Unable to determine the DB2 version as you do not have execute permission to db2level

This message is displayed if the JDBC driver is not present or the installation path to Db2 is specified incorrectly when you run the `generateDefaults` toolkit task.

There are two possible solutions to resolve this issue:

- Ensure that you provide the JDBC driver for your deployment. For more information, see [“Specifying the JDBC driver”](#) on page 68.
- Ensure that the value for the `db.installation.dir` setting in the `environment.properties` file is correct. For more information, see [environment.properties](#)

Invalid attribute value for 'name' in element 'complexType'. Recorded reason: cvc-datatype-valid.1.2.1: 'Direcci|nItemClass' is not a valid value for 'NCName'.

This message is displayed when trying to deploy with a highlight query configuration that contains an invalid character for the Db2 code page. After you install Db2, you specify the code page that the Db2 instance uses. For more information about the value that you must use for the code page, see the [Post-install section of Installing IBM Db2 for i2 Analyze](#).

Connecting clients

After you deploy and start i2 Analyze, connect to your system by using one of the supported clients and search for data. Depending on the deployed components, you can access an i2 Analyze system by using either IBM i2 Analyst's Notebook Premium or one of the web clients.

- Set up at least one user on the application server that has permission to access records in i2 Analyze.
- Ensure that i2 Analyze is started.
- Make a note of the URI for connections. When you start i2 Analyze, the URI that can be used to connect is displayed in the console. For example:

```
Web application available (default_host): http://host_name:9082/opa1
```

To use Analyst's Notebook to connect, you must also:

- Install Analyst's Notebook Premium. For more information, see the [IBM i2 Analyst's Notebook Premium](#) documentation.

To use a web client to connect, you must also:

- Ensure that you have the correct licence agreements in place to use the i2 Investigate or the i2 Notebook web client.
- If you want to use the i2 Notebook web client, ensure that you have configured your deployment to use SSL and given users the relevant command access permission. For more information, see [Enabling access to the i2 Notebook web client](#).

Note: When you create an example deployment, the example user already has the necessary permission to use the i2 Notebook web client. Also, if you have access to the i2 Analyze server, you can connect to the client through the `localhost` URL without configuring SSL.

- To use Analyst's Notebook Premium to connect, follow the instructions in the [IBM i2 Analyst's Notebook Premium](#) documentation.
- To use a web client to connect:
 - a) Open a web browser, and navigate to `https://host_name/opa1`.
The web client displays a login dialog.
 - b) Enter the name and password of a user who is registered in the application server.
If that user has permission to use the i2 Notebook web client, you see that application's user interface. If they do not have permission, you see the i2 Investigate web client user interface instead.
 - c) Search for and visualize data to verify that the application is working correctly.

Removing a deployment of i2 Analyze

You might need to reset a deployment of i2 Analyze to its just-installed state. For example, you might complete an example deployment and then want to start creating a production deployment on the same system.

Important: This procedure completely removes all the data and deployed components of i2 Analyze. If you want to retain any data from the system, back up your data first. For more information about backing up your data, see [Backing up a deployment](#).

To remove a deployment of i2 Analyze, you must remove the database from the system and then the deployment and data files from disk.

1. Stop the deployment of i2 Analyze:

- In a single-server or remote database topology, run `setup -t stop`.
 - In a multiple-server topology, see [Stopping and starting i2 Analyze](#).
2. If your deployment contains a data store, drop the database from the system:

```
setup -t dropDatabases
```

A message is displayed when you run the toolkit task to confirm that you want to complete the action. Enter Y to continue.

To ensure that the database is removed correctly, use a database management tool to try and connect to the database.

3. Move or delete the deployment and data directories in every i2 Analyze deployment toolkit in your environment.

By default, the directories are: `IBM\i2analyze\deploy` and `IBM\i2analyze\data`:

- The deployment directories for the components of i2 Analyze, and the data directory for i2 Analyze is specified in the `environment.properties` file. For more information see [environment.properties](#).
 - The data directories for the components of i2 Analyze are specified in the `topology.xml` file. For more information about this file, see [topology.xml](#).
4. Move or delete the `toolkit\configuration` directory in every i2 Analyze deployment toolkit in your environment.

The i2 Analyze deployment is returned to its just-installed state.

Best practices

When you create or manage a production deployment of i2 Analyze, there are a number of best practices that you might follow.

Server host names

When you have multiple environments with i2 Analyze deployed, it can be useful to use host names for the servers and application that identify their purpose. For example, in your development environment you might set the host names of servers as follows: `i2analyze.development.server`.

Configuration management

When you create and maintain a deployment of i2 Analyze, it is an iterative process. Because of the iterative nature, it is important to keep a record of changes to the i2 Analyze configuration. By using a source control system, you can maintain a history of changes to the configuration.

After you populate the configuration directory in the pre-production environment, make a copy of the directory in a location that is managed by a source control system.

Permanent environments

You can maintain your final development and test environments so that you can return to previous deployment phases or use the environments to test any future upgrades of the system.

Production changes

Make any configuration changes in the lowest available deployment, then promote up to production.

Installing i2 Analyze

You can install i2 Analyze by extracting an archive file. By using the archive file to install i2 Analyze, you do not need to install IBM Installation Manager on the server that you install i2 Analyze on.

For details of the system requirements, see [Software Product Compatibility Reports](#).

To install i2 Analyze, you must have the i2 Analyze version 4.3.5 distribution. Choose one of the following distributions to install i2 Analyze from:

- IBM i2 Analyze V4.3.5 (Archive install) for Windows
- IBM i2 Analyze V4.3.5 (Archive install) for Linux

Installing i2 Analyze by extracting an archive file is useful when you are installing and deploying i2 Analyze on multiple servers, or on a server without a graphical user interface.

i2 Analyze is provided in a `.zip` archive file for Windows, and a `.tar.gz` archive file for Linux. To install i2 Analyze, extract the archive file and then accept the license agreement.

1. Download the i2 Analyze distribution file for your operating system, and extract the contents into one of the following directories:

- On Windows, `C:\IBM\i2analyze`
- On Linux, `/opt/IBM/i2analyze`

The following files and directories are present in the `IBM\i2analyze` directory:

- `license`
- `swidtag`
- `toolkit`
- `license_acknowledgment.txt`

Before you can use i2 Analyze, you must read and accept the license agreement and copyright notices.

2. In a text editor, open the `notices` file and the license file for your language from the `i2analyze\license` directory.
For example, the English license is in the `LA_en` file.
3. Accept the license and copyright notices.
 - a) Open the `IBM\i2analyze\license_acknowledgment.txt` file.
 - b) To accept the license and copyright notices, change the value of `LIC_AGREEMENT` to `ACCEPT`.
For example:

```
LIC_AGREEMENT = ACCEPT
```

- c) Save and close the file.

Software prerequisites

The software prerequisites that you require depend on the deployment pattern and deployment topology of i2 Analyze that you want to deploy.

The software prerequisites that you might require are:

- A supported database management system.

- IBM Db2 Enterprise Server, Advanced Enterprise Server, Workgroup Server, or Advanced Workgroup Server editions at version 10.5 Fix Pack 10 or later and version 11.1 Fix Pack 3 or later, or Advanced and Standard editions at version 11.5.

IBM Db2 Standard Edition - VPC Option - version 11.5 is included with i2 Analyze.

For more information about how to install Db2, see [“Installing IBM Db2 for i2 Analyze”](#) on page 95.

- Microsoft SQL Server Standard or Enterprise editions at version 14.0 (2017) or 15.0 (2019).

For more information about how to install SQL Server, see [“Installing Microsoft SQL Server for i2 Analyze”](#) on page 97.

- An HTTP server that supports a reverse proxy.

If you decide to use an HTTP server, it must be configured to handle WebSocket requests.

The deployment toolkit can automatically configure an IBM HTTP Server instance on the server where the i2 Analyze application is deployed to act as a reverse proxy. To support this approach, you must install IBM HTTP Server 9.0.0.7.

For more information about how to install IBM HTTP Server, see [“Installing IBM HTTP Server for i2 Analyze”](#) on page 98.

For more information about the system requirements and prerequisites, see [Release Material](#).

Installing IBM Db2 for i2 Analyze

Location

If you are creating a production deployment, you can install Db2 in any location. When you install Db2, record the location of the installation directory because you must specify this location in the deployment toolkit before you can deploy i2 Analyze.

If you are creating an **example deployment**, install Db2 in the following location:

- For Windows: C:\Program Files\IBM\SQLLIB
- For Linux: /opt/ibm/db2/Db2_version

Features and language

In all deployments, you must ensure that the following features are installed:

- Spatial Extender server support
- Spatial Extender client

When you deploy i2 Analyze with the Information Store, you must install Db2 with the product interface language set to English only. Additionally, if you install Db2 on Red Hat® Enterprise Linux, you must use an English version of Red Hat Enterprise Linux. For more information, see [Changing the Db2 interface language \(Linux and UNIX\)](#) and [Changing the Db2 product interface language \(Windows\)](#).

For Linux deployments, if you are deploying with a schema that contains non-English characters, ensure that the operating system's LANG environment variable is set to the locale of the non-English characters.

You can only deploy i2 Analyze with a Db2 instance where the DB2_WORKLOAD environment variable is not set and the database is row-organized. If you have an existing Db2 instance where DB2_WORKLOAD is set or the database is column-organized, you must create a Db2 instance with the supported configuration and deploy i2 Analyze with it. For more information about the DB2_WORKLOAD environment

variable, see [System environment](#). For more information about column-organized databases, see [Setting the default table organization](#).

Users

On Windows, Db2 creates a Windows user account (db2admin), and two Windows groups (DB2ADMNS, DB2USERS). To work successfully with Db2, ensure that your Windows user account is a member of the DB2ADMNS Windows group.

On Linux, Db2 creates an Administration Server user (dasusr1) and group (dasadm1), an instance-owning user (db2inst1) and group (db2iadm1), and a fenced user (db2fenc1) and group (db2fadm1). To work successfully with Db2, ensure that the user that runs the deployment script is a member of the dasadm1 and db2iadm1 groups.

Make a note of any user names and passwords that are specified during the installation process.

Note: In all scenarios, the user that you use to run the deployment scripts must have permission to create and modify the database.

Post-install

After you install Db2 for the Information Store, you must enable the administrative task scheduler and set the code page on the Db2 installation:

1. On the command line, navigate to the SQLLIB\bin directory of your Db2 installation. On Linux, navigate to the db2inst1/sllib/bin directory.
2. To enable the administrative task scheduler, run the following command:

```
db2set DB2_ATS_ENABLE=YES
```

3. To set the code page for UTF-8 encoding, run the following command:

```
db2set DB2CODEPAGE=1208
```

For more information about installing Db2, see [Installing Db2 database servers](#).

Remote Db2 database storage

If you plan to deploy i2 Analyze with remote database storage, you must install Db2 on your database server, and Db2 or IBM Data Server Client on the application server. Install Db2 according to the previous instructions; if you are using IBM Data Server Client, also ensure that Spatial Extender client support is installed. For more information about IBM Data Server Client, see [Installing IBM Data Server drivers and clients](#).

The instance of Db2 or IBM Data Server Client on the application server must be the same version level as the instance of Db2 on the database server. For example, if the instance of Db2 on your database server is version 11.1, the instance of Db2 or IBM Data Server Client on the application server must also be version 11.1.

High availability

When you install Db2 in a deployment that uses HADR, the following must be true:

- The version of Db2 installed on the primary and standby servers must be the same.
- Install Db2 on all servers according to the previous information
- Db2 must be installed with the same bit size (32 or 64 bit) for both the primary and standby servers.

- The primary and standby databases must have the same database name.
- The primary and standby databases must be in different instances.
- The amount of space allocated for log files must be the same on both the primary and standby databases.

For more information about the Db2 requirements, see [High availability disaster recovery \(HADR\) support](#). For more information about the Db2 HADR feature, see [High availability disaster recovery \(HADR\)](#).

Installing Microsoft SQL Server for i2 Analyze

Location

If you are creating a production deployment, you can install SQL Server in any location. When you install SQL Server, record the location of the installation directory because you must specify this location in the deployment toolkit before you can deploy i2 Analyze.

If you are creating an **example deployment**, install SQL Server in the default location:

- For Windows: C:\Program Files\Microsoft SQL Server
- For Linux: /opt/mssql. Install the SQL Server tools in the default path: /opt/mssql-tools

Features

In all deployments, you must ensure that the following features are installed or enabled:

- Database Engine Services
- SQL Server Authentication
- TCP/IP protocol

In all deployments, you must install the ODBC Driver for SQL Server and **sqlcmd** utility on your database server.

On Windows:

- Microsoft ODBC Driver 17 for SQL Server, [Microsoft ODBC Driver 17 for SQL Server](#).
- The **sqlcmd** utility <https://docs.microsoft.com/en-us/sql/tools/sqlcmd-utility?view=sql-server-2017#download-the-latest-version-of-sqlcmd-utility>.

On Linux:

- Microsoft ODBC Driver 17 for SQL Server, [Microsoft ODBC Driver 17 for SQL Server](#).
- The SQL Server command-line tools, <https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-setup-tools?view=sql-server-2017>

You can also install Microsoft SQL Server Management Studio to administer your SQL Server installation. If you are using SQL Server on Linux, you can install SQL Server Management Studio on a Windows machine and connect to your SQL Server installation.

To create an **example deployment** on Windows, the instance name that you use must be **MSSQLSERVER**. Regardless of your operating system, the port number must be 1433.

Users

You must have an SQL Server Authentication Login that has the following permissions:

- Server Roles:
 - *dbcreator*

- *bulkadmin*, to ingest the example data. The *bulkadmin* role is not supported on Linux.
- User mappings for the *msdb* database:
 - *SQLAgentUserRole*
 - *db_datareader*

Note: In all scenarios, the user that you use to run the deployment scripts must have permission to create and modify the database.

Post-install

- Ensure that the *SQL Server Agent* service is running.
- On Windows, if you want to use the instance name to connect to SQL Server ensure that the *SQL Server Browser* service is running.

Remote SQL Server database storage

If you plan to deploy i2 Analyze with remote database storage, you must install SQL Server on the database server, and SQL Server or Microsoft Command Line Utilities 17 for SQL Server on the application server. You can install SQL Server and the Command Line Utilities according to the previous instructions.

High availability

When you install SQL Server for HADR, you must ensure that the following statements are true:

- All database servers are in the same IP address range.
- All database servers are members of the same domain for SQL Server Always On Availability Groups DNS name resolution.
- Two static TCP/IP addresses, one for the Windows Failover Cluster and one for the SQL Server Always On Availability Group. The IP addresses must be in the same range.
- The same version of Windows Server is installed.

For information about installing SQL Server with SQL Server Always On Availability Groups, see [Prerequisites, Restrictions, and Recommendations for Always On availability groups](#).

Installing IBM HTTP Server for i2 Analyze

An i2 Analyze deployment might require a reverse proxy server that routes client requests to the appropriate destination.

The deployment toolkit can automatically configure an IBM HTTP Server instance to act as a reverse proxy. To support this approach, install IBM HTTP Server as follows:

Location

Install IBM HTTP Server from an archive file in the following location:

- For Windows: C:\IBM\HTTPServer
- For Linux: /opt/IBM/HTTPServer

For more information about installing, see [Installing IBM HTTP Server](#).

The Web Server Plug-ins for WebSphere® Application Server are included in the archive installer.

Post-install

- The `bin/ikeyman` file and the `bin/gskcmd` file use the Java in your provided `$JAVA_HOME` environment variable. After you deploy i2 Analyze, you can set your `$JAVA_HOME` environment variable to the directory where the deployment toolkit installs Java.
- Ensure that Microsoft Internet Information Server is either inactive or not present on the i2 Analyze server.
- On Linux, the user that you use to run the deployment scripts must have write permissions on the `/opt/IBM/HTTPServer/conf/httpd.conf` file.

Configuring i2 Analyze

During implementation of a production deployment, you need to modify the original base deployment to match the needs of your organization. When the i2 Analyze deployment is in use, you can make further configuration changes to adjust to changing needs, and to administer the system.

Configuration sections

Configuring the i2 Analyze application

To configure i2 Analyze for your organization's requirements, you complete various activities that modify its behavior. These activities can affect aspects of the system such as authenticating and authorizing users, controlling access to features, and providing appropriate search options.

Ingesting data into the Information Store

System administrators can add data to the Information Store in bulk by following an ETL (extract, transform, load) process. To update data that was added to the Information Store earlier, they can use the same process to add it again.

Connecting to external data sources

Subject to your licensing terms, a deployment of i2 Analyze can use the i2 Connect gateway to query and retrieve data from external data sources. By implementing a connector to an external data source, you enable i2 Analyze to create and display records that represent the data in that source.

IBM i2 Analyze Schema Designer

Welcome to the IBM i2 Analyze Schema Designer documentation, where you can find information about how to use IBM i2 Analyze Schema Designer.

Common tasks

Modifying a deployed i2 Analyze schema

i2 Analyze supports a limited set of small changes to the Information Store schema in a production deployment. In general, if your changes only add to the existing schema, you can apply them without the disruption that more significant changes can cause.

Configuring user security

In a deployment of i2 Analyze, you can configure how users are authenticated and authorized with the system.

Connecting to external data sources

Subject to your licensing terms, a deployment of i2 Analyze can use the i2 Connect gateway to query and retrieve data from external data sources. By implementing a connector to an external data source, you enable i2 Analyze to create and display records that represent the data in that source.

Configuring Find Matching Records

Analysts use Find Matching Records to identify when multiple records on an Analyst's Notebook Premium chart might represent the same real-world object or relationship.

Troubleshooting and support

[i2 Enterprise Insight Analysis support page](#)

[IBM Software Support home page](#)

Configuring the i2 Analyze application

To configure i2 Analyze for your organization's requirements, you complete various activities that modify its behavior. These activities can affect aspects of the system such as authenticating and authorizing users, controlling access to features, and providing appropriate search options.

If you want to modify the configuration of an i2 Analyze system that is already in production, IBM recommends that you make your changes first in the environment that is furthest away from

production. For example, start by changing the configuration development environment, and then promote those changes through pre-production, test, and finally into production.

For more information about creating deployment environments, see [Deployment phases and environments](#).

Configuring i2 Analyze schemas

An important part of developing an i2 Analyze deployment is creating the schemas for it to use. Between them, the Information Store, gateway, and connector schemas model the data that you want to analyze in the deployment.

When you deploy a schema, you also deploy one or more *charting schemes* to accompany it. These separate files specify how entities and links and their properties appear when they are visualized on charts. To provide users with more flexibility in their investigations, you can visualize data in different ways by creating multiple charting schemes.

Any schemas that you create for the i2 Connect gateway or for individual connectors are relatively flexible, and you can modify them after deployment as you see fit. However, after an Information Store schema enters production, the changes that you can make to it are strictly limited. Therefore, it is important to ensure early in the development process that if your deployment includes an Information Store, its schema meets the requirements of the organization.

Note: In some deployments of i2 Analyze, the [Chart Store](#) replaces the Information Store. The schema and the charting scheme for the Chart Store are much smaller than their equivalents for the Information Store, but changing them in a production deployment is subject to the same limitations.

You can view, edit, and create schemas with Schema Designer. For instructions on creating and editing a schema, see the [IBM i2 Analyze Schema Designer](#) documentation.

You can learn more about deploying a custom schema at the [IBM Security Learning Academy](#).

Schema settings

When a deployment of i2 Analyze includes the i2 Connect gateway and connectors that define their own schemas, the gateway interrogates the connectors for the locations of those schemas. For schemas that are hosted on the server, the i2 Analyze application retrieves their locations from the [ApolloServerSettingsMandatory.properties](#) file.

For a deployment of i2 Analyze that includes the Chart Store or the Information Store, [ApolloServerSettingsMandatory.properties](#) must contain populated `SchemaResource` and `ChartingSchemesResource` settings that provide the locations of the schema and charting scheme files:

```
SchemaResource=  
ChartingSchemesResource=
```

Note: A single deployment of i2 Analyze cannot include both the Chart Store and the Information Store. When necessary, the Information Store subsumes the Chart Store, and the Information Store schema includes the elements that describe stored Analyst's Notebook charts.

The [ApolloServerSettingsMandatory.properties](#) files in the example configurations that include the i2 Connect gateway contain extra settings that are named `Gateway.External.SchemaResource` and `Gateway.External.ChartingSchemesResource`. However, these names are not mandatory, and they are not the only settings for gateway schemas that can appear in the file.

In general, a deployment of i2 Analyze that includes the i2 Connect gateway can have any number of gateway schemas. Every pair of schema and charting scheme files must be identified in the `ApolloServerSettingsMandatory.properties` file, using the following syntax:

```
Gateway.ShortName.SchemaResource=  
Gateway.ShortName.ChartingSchemesResource=
```

You are free to specify the *ShortName* of each pair of settings as you wish. The short name that you provide is displayed to Analyst's Notebook Premium users in the names of item types from the schema. It is also displayed to system administrators in the user interface for creating type conversion mappings. And i2 Analyze uses the short name to differentiate between any item types that have the same name in separate gateway schemas.

Modifying a deployed Chart Store schema

A deployment of i2 Analyze that includes the Chart Store (with or without the i2 Connect gateway) supports a limited set of changes to the supplied example schema. By adding property types to the "Analyst's Notebook Chart" entity type, you can enhance users' ability to categorize and retrieve the charts that they store.

The example Chart Store schema in the deployment toolkit contains a single "Analyst's Notebook Chart" entity type. That entity type contains two property types that represent the name and description of any chart that is stored in the Chart Store. In a live deployment, i2 Analyze permits additive changes to the Chart Store schema. By adding property types to the "Analyst's Notebook Chart" entity type, you can enable users to search for charts more effectively, and to filter search results by the values of the new properties.

Note: In a deployment of i2 Analyze that includes the Information Store, the schema includes the Chart Store's "Analyst's Notebook Chart" entity type alongside all the Information Store item types. The instructions here apply to modifications to that entity type in Chart Store and Information Store schemas.

If you follow this procedure in a deployment that provides high availability, you must complete each step and run each command on every Liberty server in your environment before you move to the next step or command.

The following steps describe how to add property types to the schema in a deployment of i2 Analyze that includes the Chart Store.

1. Locate the XML file that contains the Chart Store schema for the i2 Analyze deployment, and load it into Schema Designer.
2. Make your additions to the "Analyst's Notebook Chart" entity type, and then save the file.

Note: Schema Designer does not validate whether your changes are compatible with the deployed schema. Validation takes place when you apply the changes to your deployment.

3. Run the following commands on the Liberty server to update the database and application to conform to the updated schema.

```
setup -t stopLiberty  
setup -t updateSchema  
setup -t deployLiberty  
setup -t startLiberty
```

The command recognizes that you modified the schema, determines whether the changes are valid for the running Chart Store, and then applies them. If the changes are not valid, the command displays messages to explain the problems.

By default, deployments of i2 Analyze that include the Chart Store do not specify a results configuration file. Any property types that you add are automatically available as filters for search results. However, if you add a property type that you do not intend to use for filtering, or if you have modified the "Analyst's Notebook Chart" entity type in an Information Store schema, you must set up search results filtering to include or exclude the new property types as required.

When you complete the procedure, reconnect to i2 Analyze from Analyst's Notebook Premium to confirm that the changes are present and behaving correctly.

Modifying a deployed Information Store schema

i2 Analyze supports a limited set of small changes to the Information Store schema in a production deployment. In general, if your changes only add to the existing schema, you can apply them without the disruption that more significant changes can cause.

After you deploy i2 Analyze with an Information Store, you can generally make additive changes to the Information Store schema, but not destructive ones. For example, you can add new item types, and add new property types to existing item types, but you cannot remove types from the schema. For more information about the changes that you can make, see "Permitted Information Store schema changes" on page 105.

Destructive schema changes include removing item types or property types. Removing types can result in an Information Store with data that is not valid according to the schema. To make destructive changes to the schema, you must remove any data from the system and re-create its databases. For more information about performing destructive schema changes or replacing the schema, see "Replacing the Chart Store or Information Store schema" on page 106.

If you follow this procedure in a deployment that provides high availability, you must complete each step and run each command on every Liberty server in your environment before you move to the next step or command.

The following steps describe how to make small changes to the schema in a production deployment of i2 Analyze that includes an Information Store.

1. Locate the XML file that contains the Information Store schema for the i2 Analyze deployment, and load it into Schema Designer.
2. Make your changes to the schema and its associated charting schemes, and then save the file.

Note: Schema Designer does not validate whether your changes are compatible with the deployed schema. Validation takes place when you apply the changes to your deployment.

3. Run the following commands on the Liberty server to update the database and application to conform to the updated schema.

```
setup -t stopLiberty
setup -t updateSchema
setup -t deployLiberty
setup -t startLiberty
```

The command recognizes that you modified the schema, determines whether the changes are valid for the running Information Store, and then applies them. If the changes are not valid, the command displays messages to explain the problems.

Note: If you customized the Information Store creation process by specifying `createdatabase=false` in the topology file and running the scripts yourself, this command works in the same way. Execution stops so that you can customize the changes to the Information Store. After you apply the changes, you can run the task again to complete the process.

Because you can make only additive changes to an Information Store schema that you modify through this procedure, it is not mandatory to change other parts of your deployment. However, to take full advantage of your additions, consider the following complementary changes.

- To enable the Information Store to ingest data for the new item types and property types, modify your ingestion artifacts. See [Information Store data ingestion](#).
- If your deployment includes definitions of the property values of merged i2 Analyze records, you must update your merged property values definition views. See [Define how property values of merged records are calculated](#).
- If you want users to see the new types in quick search filters, edit the configuration file that controls them. See [Setting up search results filtering](#).
- If your deployment includes highlight queries, you can update them for the new item and property types. For more information, see [“Deploying highlight queries” on page 182](#).

Permitted Information Store schema changes

After you create an Information Store schema and use it in a deployment, i2 Analyze restricts subsequent changes to that schema to ensure that data in the system remains accessible. In general, you can add content to a published Information Store schema, but you cannot take content away.

To be specific, i2 Analyze prevents changes to a schema that might invalidate data that is already stored. For example, if you remove an item type from the Information Store schema, then the store might contain data for which i2 Analyze no longer has a definition. However, you can make additive changes to the schema. You can also disable or hide item or property types in some parts of the system, if you are certain that they are no longer required.

Permitted changes

You can make the following changes to the Information Store schema of a live deployment:

- Add an item type.
- Change the display name of an item type.
- Change the icon of an item type.
- Add a property type.
- Change the display name of a property type.
- Change the display order of a property type.
- Increase the value length of a property type.
- Change the logical type of a property type from a selected-from to suggested-from list.
- Make a property type non-mandatory.
- Add a grade type.
- Add a labeling scheme.

Prevented changes

i2 Analyze prevents all of the following Information Store schema changes from taking place against a live deployment:

- Change the schema identifier.
- Remove an item type.
- Remove an entity type from the permitted list for a link type.
- Remove a property type.

- Make a property type mandatory.
- Remove a default property value.
- Remove a property value from a selected-from or suggested-from list.
- Change the logical type of a property type. Except for the permitted change described previously.
- Remove a grade type.
- Remove a link strength.

To protect your data when you redeploy with a modified Information Store schema, i2 Analyze carries out validation checks to ensure that the changes you made do not result in data loss.

Permitted changes that do not affect the Information Store

You can make the following changes to the Information Store schema of a live deployment. However, they do not affect the store, which means that users still see items with hidden types when they run a quick search, for example.

- Add a link constraint.
- Add a link strength.
- Disable an item type.
- Hide an item type.

The disable and hide functions change how item and property types behave in the deployment without making any destructive changes. Use these features with caution though because they can affect the behavior of visual query and import operations.

Replacing the Chart Store or Information Store schema

The schema is a key component of any i2 Analyze deployment that includes the Chart Store or the Information Store, and configuring it correctly is an important part of the development process. Replacing or making destructive changes to the schema requires you to clear and repopulate the store. As you do so, you might need to change other parts of the system too.

This procedure contains the steps for making destructive changes to a Chart Store or Information Store schema in your deployment of i2 Analyze. When you replace the schema, you must clear data from the system, modify any sample data, and edit any other part of the configuration that depends on the schema.

In a deployment that provides high availability, use the documentation from your database management system to remove the Chart Store or Information Store database from each database server in your deployment instead of the `dropDatabases` toolkit task.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `deleteSolrCollections` and `createSolrCollections` on one Liberty server only.

1. Copy the new schema and charting scheme files to the `configuration\fragments\common\WEB-INF\classes` directory of the deployment toolkit.
2. Specify the schema and charting scheme that the deployment uses.
 - a) Using a text editor, open the `ApolloServerSettingsMandatory.properties` file in the same directory as the schema files.
 - b) Set the values of the `SchemaResource` and `ChartingSchemesResource` properties to the names of your schema and charting scheme files.
 - c) Save and close the file.

The following steps update your deployment with the new schema.

3. Stop the deployment:

```
setup -t stopLiberty
```

4. To remove the database and Solr collections, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -t dropDatabases --hostname liberty.host-name  
setup -t deleteSolrCollections --hostname liberty.host-name
```

Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

A message is displayed when you run each task to confirm that you want to complete the action. Enter `Y` to continue. The database and Solr collections are removed from the system.

5. To re-create the Solr collections and databases, run the following commands:

```
setup -t createSolrCollections --hostname liberty.host-name  
setup -t createDatabases
```

6. Update the i2 Analyze application:

```
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

If you are using a results configuration file, configure the facets to match the item and property types that you added or changed in the Information Store schema. See [“Setting up search results filtering”](#) on page 197.

If you are defining the property values of merged i2 Analyze records, you must update your merged property values definition views. See [Define how property values of merged records are calculated](#).

Adding, removing, modifying gateway schemas

A single deployment of i2 Analyze that includes the i2 Connect gateway can contain several gateway schemas. When you add or remove gateway schemas, you must edit the `ApolloServerSettingsMandatory.properties` file. It is likely that you must edit some of the other configuration files and settings too.

When several connectors query the same data source, it can make sense for them to share a gateway schema instead of defining schemas of their own. If you buy or develop more than one set of such connectors, then it is possible for your deployment of i2 Analyze to use several gateway schemas. The section of the settings file then looks like this:

```
Gateway.SourceOne.SchemaResource=  
Gateway.SourceOne.ChartingSchemasResource=  
Gateway.SourceTwo.SchemaResource=  
Gateway.SourceTwo.ChartingSchemasResource=
```

For each gateway schema that you add to a deployment, you must add two entries to this list with a new name in place of *SourceOne* or *SourceTwo*. If you remove a gateway schema, make sure to remove both entries that were associated with that schema.

If you follow this procedure in a deployment that provides high availability, you must complete each step and run each command on every Liberty server in your environment before you move to the next step or command.

After you add or remove the settings for a particular gateway schema from `ApolloServerSettingsMandatory.properties`, check other aspects of your system configuration.

1. Before you reload the i2 Analyze server to reflect your changes, make sure that the results configuration file matches your new set of gateway schemas.
 - a) Open the file indicated by the `ResultsConfigurationResource` property in `DiscoServerSettingsCommon.properties` and follow the instructions in [“Setting up search results filtering” on page 197](#) to remove references to item types from removed gateway schemas.
 - b) Add or edit elements in the file to enable result filtering on item and property types in any new gateway schemas.
2. Follow the instructions in [Modifying and testing a connector](#) to reload the server and update its information about the gateway schemas.
3. Follow the instructions in [“Configuring matching” on page 201](#) to make any changes to the system or Find Matching Records match rules files that are required as a result of your changes to the configured schemas.
4. If your deployment uses them, update the set of type conversion mappings.
 - a) Open a web browser and navigate to the i2 Analyze server admin console, as described in [Modifying and testing a connector](#).
 - b) In the **i2 Analyze type conversion** app, add, edit, or remove type conversion mappings to reflect the new set of gateway schemas.
 - c) Export the modified conversion mappings. Copy the resulting `mapping-configuration.json` file to the deployment toolkit's `configuration\fragments\common\WEB-INF\classes` directory.
 - d) [Redeploy Liberty](#) to update the type conversion mappings on the server.

On completion of the above steps, your i2 Analyze deployment is fully updated in response to the changes that you introduced by adding, removing, or modifying a gateway schema.

Configuring the security schema

After you deploy an i2 Analyze security schema, there are restrictions on the changes that you can make to the security schema. The actions that you perform to update a deployed security schema depend on the changes that you make.

You are relatively free to create security groups and permissions, add security dimension values to existing security dimensions, and change any display names. For more information about performing these types of change, see [“Modifying security permissions” on page 110](#) and [“Modifying security dimensions” on page 109](#).

If you want to replace the security schema, make destructive changes, and add security dimensions, you must remove and re-create the underlying database in the system. For more information about performing these types of change, see [“Replacing a security schema” on page 113](#).

Important: To change the schema in the way that this documentation describes, your deployment must use the supplied `WebSphereDynamicAccessRoleBasedPrincipalProvider` security implementation.

Modifying security dimensions

You can add dimension values to security dimensions, and you can make cosmetic changes to dimensions and values without clearing your data from the system.

The following table shows the changes that you can make to the security dimensions in a deployed security schema without clearing data from the system. It also states whether a reindex is required if the change takes place:

Change	XML elements or attributes	Allowed	Reindex required
Add a security dimension	<Dimension>	No	N/A
Modify an existing security dimension	DisplayName, Description	Yes	No
Remove an existing security dimension	<Dimension>	No	N/A
Add a dimension value to an unordered security dimension	<DimensionValue>	Yes	No
Add a dimension value to an ordered security dimension	<DimensionValue>	Yes	Yes ¹
Reorder the dimension values in an ordered security dimension	<DimensionValue>	Yes	Yes
Modify an existing dimension value	DisplayName, Description	Yes	No
Remove an existing dimension value from a security dimension	<DimensionValue>	No	N/A

¹ If you add a dimension value to the bottom of the sequence in an ordered security dimension, you do not need to complete a reindex.

To add a security dimension value to a security dimension, add a <DimensionValue> element as a child of an existing <Dimension> element.

To modify the display name or description of a dimension or a dimension value, change the DisplayName or Description attributes an existing <Dimension> or <DimensionValue> element. You must not change the value of the Id attribute.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using an XML editor, open the security schema for the deployment.

The security schema is in the toolkit\configuration\fragments\common\WEB-INF\classes\ directory. The name of the security schema is specified in the DynamicSecuritySchemaResource property of the ApolloServerSettingsMandatory.properties file in the same directory.

2. Modify the security dimensions in the security schema according to your requirements.
3. Increment the version number that is stated in the Version attribute of the <SecurityDimensions> element in the security schema.
4. Check your updated schema to ensure that it remains possible for all users to get an access level that is not "none" for at least one value in every access dimension.
5. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

6. In a command prompt, navigate to the toolkit\scripts directory.

7. Stop Liberty:

```
setup -t stopLiberty
```

8. If you completed a change that requires a reindex, clear the search index:

```
setup -t clearSearchIndex --hostname liberty.hostname
```

In a deployment that provides high availability, you only need to run this command on one Liberty server.

9. Update and redeploy the system:

```
setup -t updateSecuritySchema  
setup -t deployLiberty
```

10. Start Liberty:

```
setup -t startLiberty
```

Modifying security permissions

It is possible to change the mapping between user groups and the security permissions that the security schema defines without reimporting or reindexing your data. You must take care to ensure that all i2 Analyze users retain the ability to access your deployment.

The following table shows the changes that you can make to the security permissions in a deployed security schema without clearing data from the system:

Change	XML elements or attributes	Allowed	Reindex required
Add a security group	<GroupPermissions>	Yes	No
Modify an existing security group	UserGroup	Yes	No
Remove an existing security group	<GroupPermissions>	Yes	No
Add security dimensions to a security group	<Permissions>	Yes	No
Remove security dimension from a security group	<Permissions>	Yes	No
Add security permissions from a security dimension for a security group	<Permission>	Yes	No
Modify existing security level from a security dimension permission for a security group	DimensionValue, Level	Yes	No
Remove existing security permissions from a security dimension permissions element for a security group	<Permission>	Yes	No

If the requirements for security groups change, you can modify the <GroupPermissions> element and its children.

- To add a group, insert a complete <GroupPermissions> element. To use the new group, you must ensure that the user repository contains a group that matches the value of the UserGroup attribute.
- To modify the name that is associated with a group, change the value of the UserGroup attribute.

- To remove a group, remove the <GroupPermissions> element for that group.

If the requirements for the permissions of a security group change, you can add or remove <Permissions> elements, and add, modify, and remove child <Permission> elements.

- To change the dimensions that a group has permissions for, you can add or remove <Permissions> elements as follows:
 - To add a dimension that the group has permissions for, insert a <Permissions> element where the value of the Id attribute matches the value of the Id attribute of the dimension.
 - To remove a dimension that the group has permissions for, remove the <Permissions> element where the value of the Id attribute matches the value of the Id attribute of the dimension.
- To change the security permissions that a group has within a dimension, you can add, modify, and remove <Permission> elements as follows:
 - To add a permission to a group, insert a <Permission> element. The DimensionValue attribute must match a dimension value in the same dimension that is defined in the Dimension attribute of the parent <Permissions> element.
 - To modify the current permission that a group has in a dimension value, set the Level attribute to a different value.
 - To modify the dimension value that a permission is for, set the DimensionValue attribute to a different value.
 - To remove the current permission that a group has in dimension value, remove the <Permission> element in which the DimensionValue attribute matches that dimension value.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using an XML editor, open the security schema for the deployment.

The security schema is in the toolkit\configuration\fragments\common\WEB-INF\classes\ directory. The name of the security schema is specified in the DynamicSecuritySchemaResource property of the ApolloServerSettingsMandatory.properties file in the same directory.

2. Modify the security permissions in the security schema according to your requirements.
3. Increment the version number that is stated in the Version attribute of the <SecurityDimensions> element in the security schema.
4. Check your updated schema to ensure that it remains possible for all users to get an access level that is not "none" for at least one value in every access dimension.
5. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

6. In a command prompt, navigate to the toolkit\scripts directory.
7. Stop Liberty:

```
setup -t stopLiberty
```

8. Update and redeploy the system:

```
setup -t updateSecuritySchema  
setup -t deployLiberty
```

9. Start Liberty:

```
setup -t startLiberty
```

Setting default dimension values

When users create i2 Analyze records in Analyst's Notebook Premium, i2 Analyze applies a default set of dimension values to the record. Similarly, when the Information Store ingests data, the created records receive dimension values during the process.

You might change the default security dimension values that are applied to records for different reasons:

- If you change the dimensions or dimension values in your security schema.
- If the security requirements of your deployment change.

Regardless of your reason for changing the default security dimension values, or when the values are applied, each record must have at least one dimension value from each security dimension.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using an XML editor, open the security schema for the deployment.

The security schema is in the `toolkit\configuration\fragments\common\WEB-INF\classes\` directory. The name of the security schema is specified in the `DynamicSecuritySchemaResource` property of the `ApolloServerSettingsMandatory.properties` file in the same directory.

2. Update the value of the `<DefaultSecurityDimensionValues>` element with a comma separated list of identifiers of security dimension values that you want to be applied by default.

For example, `<DefaultSecurityDimensionValues>CON,OSI,HI</DefaultSecurityDimensionValues>`.

In a standard deployment of i2 Analyze, a supplied implementation of the `DefaultSecurityDimensionValuesProvider` that applies dimension values from the `<DefaultSecurityDimensionValues>` element is used. You can create your own implementation that does not use these values by using i2 Analyze Developer Essentials.

3. When data is ingested into the Information Store, the records that it contains receive their security dimension values during the ingestion process.

To change the behavior of the ingestion process, update the contents of the `<securityDimensionValues>` element in the mapping file with the identifiers of security dimension values that you want to be applied by default. For more information about updating the default security dimension values that are applied during ingestion, see [Information Store data ingestion](#).

4. In a command prompt, navigate to the `toolkit\scripts` directory.
5. Stop Liberty:

```
setup -t stopLiberty
```

6. Update the i2 Analyze application:

```
setup -t updateSecuritySchema
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

Replacing a security schema

The security schema is a key component of an i2 Analyze deployment, and configuring it correctly is an important part of the development process. Replacing or making destructive changes to the security schema requires you to clear and repopulate the data stores in your deployment.

In a deployment that provides high availability, use the documentation from your database management system to remove the Information Store database from each database server in your deployment instead of the `dropDatabases` toolkit task.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `deleteSolrCollections` and `createSolrCollections` on one Liberty server only.

1. Modify or create the security schema that you want to update your deployment with.

For more information about creating the security schema, see [“Creating a security schema” on page 114](#).

2. Update the configuration with your security schema.

- a) Ensure that the security schema file is in the `configuration\fragments\common\WEB-INF\classes` directory.

- b) Ensure that your security schema file is specified in `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties`.

- c) Ensure that the identifiers of the security dimension values that records receive by default are valid in the `<DefaultSecurityDimensionValues>` element in your security schema.

For more information, see [“Setting default dimension values” on page 112](#).

The following steps update your deployment with the new security schema.

3. Stop the deployment:

```
setup -t stopLiberty
```

4. To remove the database and Solr collections, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -t dropDatabases --hostname liberty.host-name  
setup -t deleteSolrCollections --hostname liberty.host-name
```

Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

A message is displayed when you run each task to confirm that you want to complete the action. Enter `Y` to continue. The database and Solr collections are removed from the system.

5. To re-create the Solr collections and databases, run the following commands:

```
setup -t createSolrCollections --hostname liberty.host-name  
setup -t createDatabases
```

6. Update and redeploy the system:

```
setup -t updateSecuritySchema  
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

8. If you changed the names of the user groups in the security schema, update the basic user registry to match the new names.

For more information, see [Configuring the Liberty user registry](#).

Add some data to the system, and verify that users see the behavior that you intended. Iterate over the process of modifying and replacing the schema as many times as you need.

The i2 Analyze security schema

An i2 Analyze security schema defines the security dimension values that can be assigned to records, and the security permissions that you assign to groups of users. You must develop a security schema that meets the requirements of your organization.

The fact that the Information Store contains dimension values means that significant changes to the security schema can require you to clear and reingest or repopulate data. This behavior is reasonable while you develop your deployment, but not after you deploy into production. As a result, it is important to refine the security schema before you put it into production.

Creating a security schema

Every deployment of i2 Analyze requires a security schema that encapsulates the security model for that deployment. The easiest way to create a security schema is to start from the example that IBM provides with the platform.

Before you create the XML security schema file, you must design the security model for your deployment of i2 Analyze. In particular, you must identify or create the user groups to which security permissions are assigned.

When you deploy i2 Analyze, the group names in your security schema must match the names of user groups in your user repository.

An i2 Analyze security schema contains definitions of security dimensions and security permissions. When you create a security schema, you define the dimensions and dimension values first, and then define the security permissions that refer to them.

1. Navigate to the directory in the deployment toolkit that contains the example security schema:
toolkit\configuration\examples\security-schema\example-dynamic-security-schema.xml.
2. Make a copy of the example-dynamic-security-schema.xml file, give it an appropriate name, and then open it in an XML editor.
3. Edit the contents of the <AccessSecurityDimensions> element so that it contains a <Dimension> element for each category that your deployment uses to determine access rights to items and records in i2 data stores.
4. Edit the contents of the <SecurityPermissions> element:
 - a) Add or modify <GroupPermissions> elements so that they reflect all the user groups to which you assign security permissions. The group names in your security schema must match the names of user groups in the user repository.
 - b) Within each <GroupPermissions> element, add or modify <Permissions> elements to indicate which dimensions are affected by membership of each user group.
 - c) Within each <Permissions> element, add or modify <Permission> elements to assign security levels to items and records that have particular dimension values.

There are three permitted values for the Level attribute of the <Permission> element:

- NONE

- READ_ONLY
 - UPDATE
5. Edit the contents of the <DefaultSecurityDimensionValues> element to define the default security dimension values that i2 Analyze provides to records that users create in Analyst's Notebook Premium.
Update the value of the <DefaultSecurityDimensionValues> element with a comma separated list of identifiers of security dimension values that you want to be applied by default. For example, <DefaultSecurityDimensionValues>CON,OSI,HI</DefaultSecurityDimensionValues>.
 6. Save the completed security schema to the configuration\fragments\common\WEB-INF\classes directory in the deployment toolkit.

Security schemas

An i2 Analyze security schema defines the security dimensions that exist in a deployment, and the dimension values that can be assigned to items and records. A security schema also defines the permissions that i2 Analyze users can receive.

Every deployment of i2 Analyze has a security schema whose contents reflect local requirements. It is the responsibility of the deployer to ensure that the security schema is appropriate for the environment where it is used. Often, the security dimensions map to security classifications that exist in the organization.

Before you create a security schema, it is important to understand the relationship between the security model and the security schema. For more information, see [IBM i2 Analyze security model](#) .

Security dimensions

A security schema defines access security dimensions that contain dimensions and dimension values.

Security permissions

A security schema defines security permissions by user group, and then by dimension. For a particular user group, the schema identifies one or more dimensions for which membership of that group affects access rights. For each identified dimension, the schema contains a list of security permissions.

It is not necessary for the security schema to define permissions for every user group in the organization. Similarly, it is not necessary for the permissions within any particular dimension or group to set a security level for every possible dimension value. The completeness of the schema is judged at run time when the security level of a particular user for a particular item or record is calculated.

Security schema definitions

An i2 Analyze security schema is an XML file with a relatively simple structure. Security dimensions and security permissions are defined in separate sections of the file.

In outline, the <SecuritySchema> root element of a security schema contains child elements for the dimension and permission definitions:

```
<SecuritySchema>
  <SecurityDimensions Id="" Version="">
    <AccessSecurityDimensions>
      <Dimension ...>
        <DimensionValue ... />
        ...
      </Dimension>
      ...
    </AccessSecurityDimensions>
  </SecurityDimensions>

  <SecurityPermissions>
    <GroupPermissions ...>
      <Permissions ...>
        <Permission ... />
        ...
      </Permissions>
      ...
    </GroupPermissions>
    ...
  </SecurityPermissions>
</SecuritySchema>
```

The <SecurityDimensions> element has attributes for the Id and Version of the schema. If you modify any part of the security schema, retain the identifier but increment the version number. In this way, you ensure that all i2 data stores and services are informed of the changes.

In a valid security schema, the <AccessSecurityDimensions> element must be present, and there must be at least one <GroupPermissions> element inside <SecurityPermissions>.

Security dimension definitions

Security dimensions are defined in an i2 Analyze security schema file as children of the mandatory <AccessSecurityDimensions> element. A valid security schema defines at least one access security dimension.

The following example shows a simple, complete <AccessSecurityDimensions> element:

```
<AccessSecurityDimensions>
  <Dimension Id="SD-SC"
    DisplayName="Security Classification"
    Description="The security classification of this information"
    Ordered="true">
    <DimensionValue Id="TOP" DisplayName="Top Secret" Description="Top Secret" />
    <DimensionValue Id="RES" DisplayName="Restricted" Description="Restricted" />
  </Dimension>
</AccessSecurityDimensions>
```

The attributes of the <Dimension> element affect how the values in the security dimension are interpreted.

Attribute	Description
Id	A unique identifier that is used to distinguish this security dimension throughout the system.
DisplayName	A name that identifies this dimension to the user in clients.
Description	A more detailed description of this security dimension that provides more information to the user. In the Analyst's Notebook Premium, the description is used as a tooltip.
Ordered	Indicates whether the values in this dimension form a descending sequence in which each value supersedes the values below it. Marking this dimension as Ordered="true" means that a user who has access rights to "Top Secret" data implicitly has the same access rights to "Restricted" data as well. For a dimension in which Ordered="false", there is no such implication, and access rights must be assigned explicitly for each dimension value.

The Id, DisplayName, and Description attributes of <DimensionValue> elements have the same purpose and meaning as the <Dimension> attributes with the same names. The identifiers of dimension values must be unique within the dimension that defines them.

Important: After you deploy i2 Analyze, the changes that you can make to security dimensions are limited. You cannot add or remove dimensions, or remove dimension values. You can only add values to existing dimensions. For this reason, you must understand the requirements of your organization before you deploy i2 Analyze in a production environment.

Security group permission definitions

In an i2 Analyze security schema, the mandatory <SecurityPermissions> element contains one or more <GroupPermissions> elements. Each <GroupPermissions> element defines the security levels that users in a particular group receive for items and records with particular dimension values in an i2 Analyze deployment.

The syntax for defining the security permissions for user groups enables membership of one group to convey permissions across several dimensions, and allows different groups to convey different permissions for the same dimensions. The following example shows how to structure <GroupPermissions> elements inside the <SecurityPermissions> element:

```

<SecurityPermissions>
  <GroupPermissions UserGroup="Clerk">
    <Permissions Dimension="SD-SC">
      <Permission ... />
    </Permissions>
    ...
  </GroupPermissions>
  <GroupPermissions UserGroup="Manager">
    <Permissions Dimension="SD-SC">
      <Permission ... />
      ...
    </Permissions>
    <Permissions Dimension="SD-IT">
      <Permission ... />
      ...
    </Permissions>
    ...
  </GroupPermissions>
  <GroupPermissions UserGroup="Security Controller">
    <Permissions Dimension="SD-GA">
      <Permission ... />
    </Permissions>
  </GroupPermissions>
</SecurityPermissions>

```

The value of the UserGroup attribute of each <GroupPermissions> element must match the name of a group of i2 Analyze users.

The value of the Dimension attribute of each <Permissions> element must match the identifier of one of the dimensions that is defined in the first part of the schema.

It is normal for <Permissions> elements for the same dimension to appear in more than one <GroupPermissions> element:

- Users who are members of one group but not the other can receive different access levels on items and records that have the same dimension values.
- When users are members of more than one group, <Permissions> elements for the same dimension are combined before any access level calculation takes place.

Important: You can add and remove <GroupPermissions> elements from a deployed security schema, if the resulting system continues to obey the rules of i2 Analyze. In particular, it must remain possible for all users to get an access level that is not "none" for at least one value in every access dimension.

Security permission definitions

The security permission definitions in an i2 Analyze security schema each associate a single dimension value with a single security level. The definitions can be simple because of the additional context that their location in the security schema file provides.

The <Permission> elements that define security permissions always appear inside <Permissions> elements, which in turn always appear inside <GroupPermissions> elements.

```

<GroupPermissions UserGroup="Manager">
  <Permissions Dimension="SD-SC">
    <Permission DimensionValue="TOP" Level="UPDATE" />
    <Permission DimensionValue="RES" Level="UPDATE" />
  </Permissions>
  <Permissions Dimension="SD-IT">
    <Permission DimensionValue="HUMINT" Level="READ_ONLY" />
  </Permissions>
</GroupPermissions>

```

It is possible, and often desirable, for identical `<Permission>` elements to appear in different locations in an i2 Analyze security schema. The effect of a security permission definition depends entirely on its position in the file.

Important: Like the `<GroupPermissions>` elements that contain them, you can add and remove `<Permissions>` and `<Permission>` elements from a deployed security schema, if the resulting system does not break the rules of i2 Analyze.

Item type security

In some deployments of i2 Analyze, it is necessary to configure the system so that only users in specific groups are able to interact with records with particular item types. For users without the necessary permissions, records with those types are invisible - it is as if they do not exist.

The `type-access-configuration.xml` file in the `toolkit/configuration/live` directory contains the item type security configuration, which defines how user groups have visibility of specified item types.

Note: The `type-access-configuration.xml` file allows item type security to be determined by user group membership. However, it is possible to configure item type security based on additional user information, for example their record security access permissions (defined by the security schema). To do this, you must implement the [item type security SPI](#).

By default (and ignoring metadata), the `type-access-configuration.xml` file contains only its root element:

```

<tns:TypePermissions>
</tns:TypePermissions>

```

If the file does not exist or remains in its default state, then all users have access to all records, subject to the rules of the security schema.

Note: The configuration file is processed after item type conversion takes place. As such, the restrictions you define apply only to the types that are present post-mapping.

Configuration example

Given a deployed schema that contains the entity types ET1, ET2, ET3, and LT1, consider the following configuration:

```
<tns:TypePermissions DefaultSchemaShortName="...">
  <ItemType Id="ET1" SchemaShortName="...">
    <Allow>
      <UserGroup Name="Analyst"/>
      <UserGroup Name="Clerk"/>
    </Allow>
  </ItemType>
  <ItemType Id="ET3">
    <Allow></Allow>
  </ItemType>
  <ItemType Id="LT1"></ItemType>
</tns:TypePermissions>
```

This configuration defines the following:

1. Records with type ET1 are visible only to users who belong to the Analyst or Clerk groups. Users belonging to other groups are not able to see records with this type.
2. There is no permission for the type ET2. All users can see records with that type.
3. The permission for type ET3 contains an empty <Allow> element. This means that records with type ET3 are invisible to *all* users except those in groups that have the i2:Administrator command access permission.
4. The permission for type LT1 contains *no* <Allow> element. As such, just like ET2, all users can see records with the LT1 type.

Schema short names

In a deployment of i2 Analyze that contains several schemas, it is possible for item types from different schemas to have the same identifier. To avoid this potential problem, you can specify the short name of the schema that contains each type you want to configure.

As shown in the example above, you specify the schema short name through either the SchemaShortName attribute on an <ItemType> element, or the DefaultSchemaShortName attribute on the root <tns:TypePermissions> element.

If you omit the SchemaShortName attribute for an <ItemType> element, the value of DefaultSchemaShortName is used instead. If you provide neither attribute, i2 Analyze attempts to resolve the identifier against all schemas in the deployment. If the item type appears in more than one schema (or in none of them), i2 Analyze logs a warning that it cannot resolve the specified item type.

Note: It is not possible to specify more than one <ItemType> element with the same Id and SchemaShortName attributes.

Other considerations

Take note of the following considerations when you configure item type security in your deployment of i2 Analyze.

Command access control

Users in groups with the `i2:Administrator` command access permission are never affected by item type permissions.

Connectors & services

A connector service is hidden from the list of services on the client if *all* the possible item types of result records, as defined by `resultItemTypes` in the service configuration, are invisible to the user. If all of a connector's services are hidden for this reason, the connector itself is hidden.

Entities & links

Item type security restrictions on one entity type have no implications for other entity types. Similarly, restrictions on one link type have no implications for other link types, or for entity types. However, restrictions on the visibility of entity records *can* also affect the visibility of link records.

Taking the example configuration above, the deployed schema might specify that the entity type ET1 is the *only* permitted type for records at one end of links with type LT1. In other words, ET1 is the only entity type identifier in the `fromTypeIds` or `toTypeIds` attribute for the link type, like this:

```
<LinkType Id="LT1" FromTypeIds="ET1" ToTypeIds="ET2" ...>
```

In this situation, users who cannot see records of type ET1 also cannot see records of type LT1.

On the other hand, if the definition of LT1 specifies other valid link end types that are not restricted for the same users, as in this case, then records of type LT1 remain visible:

```
<LinkType Id="LT1" FromTypeIds="ET1 ET2" ToTypeIds="ET2" ...>
```

Seeds

Item type security restrictions can affect the behavior of seeded search services. The following rules apply to services configured with `seedConstraints`:

- If the service accepts seeds of a restricted type, and does not specify that it must have at least one seed of that type (that is, it is possible to use the service without a seed of the restricted type), then *that constraint* is removed from the user's view.
- If the service accepts seeds of a restricted type, and it must have at least one seed of that type, then *the service* is removed from the user's view.

Item type conversion

As previously mentioned, item type restrictions apply only after item type conversion has taken place.

Asynchronous queries

Asynchronous queries are validated when they are launched, to ensure that the seeds are visible to the user.

If a user's permissions change after initiating the asynchronous query on the connector, the query runs to completion, but any results no longer visible to the user are filtered out of the result set (which might be empty as a result).

Changing item type permissions

By default, deployments of i2 Analyze do not define item type permissions. All users can see records of all types, provided that per-record security does not prevent them from doing so. To define or subsequently modify item type permissions, you must edit and redeploy the type access configuration.

Before you begin

The type access configuration is one of a group of configuration files that you can modify and send to the i2 Analyze server without the need for system downtime. These files are stored in the `toolkit/configuration/live` directory.

To enable updating the server without the need to restart it, ensure that you have access to a command-line tool such as `postman` or `curl`.

Procedure

Edit the configuration file:

1. If you do not already have one, obtain and configure an XML editor.
2. In the XML editor, open the `toolkit/configuration/live/type-access-configuration.xml` file.
3. Using the [reference](#) and [example](#) information, modify the file to define the type access permissions that you need.

Update the deployment with your changes.

The following method deploys your changes without stopping the server, through a POST request to a REST endpoint.

To redeploy your changes using only the deployment toolkit, see [Redeploying Liberty](#). You must use the deployment toolkit if you are in a deployment with high availability, or if you are deploying to your production environment.

1. At the command line, navigate to the `toolkit/scripts` directory.
2. Update the server with your configuration file:

```
setup -t updateLiveConfiguration
```

3. Update the running application by using the `reload` endpoint. Make sure that you provide the credentials of a user with administration rights:

```
curl -i --cookie-jar cookie.txt -d j_username=<user_name>
      -d j_password=<password>
      http://<host_name>/<context_root>/j_security_check

curl -i --cookie cookie.txt -X
      POST http://<host_name>/<context_root>/api/v1/admin/config/reload
```

Warning: `reload` updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

The server validates the item type configuration as it loads, and returns any errors in its response.

Test the new and updated item type permissions.

1. A good way to verify that your item type security configuration is loaded correctly is to call connector or gateway schema endpoints and search the response body for an item type that the current user should not be able to see.

The item type security configuration file

The `type-access-configuration.xml` file controls which users get access to each of the item types declared in the schema. The set of permitted XML elements in an item type security configuration file is relatively small.

Root element

`<TypePermissions>`

`<TypePermissions>` is the root element of the item type security configuration file. In the file that deployments of i2 Analyze receive by default, the element is empty and its name is prefixed with the `tns` namespace:

```
<tns:TypePermissions DefaultSchemaShortName="...">
  ...
</tns:TypePermissions>
```

The `<TypePermissions>` element has a single, optional attribute named `DefaultSchemaShortName`. If the item types that you want to constrain are all or mostly defined in a particular schema, then it is efficient to provide the short name of that schema here.

When the `<TypePermissions>` element is empty, there are no item type constraints on the records that users can see.

Item type elements

`<ItemType>`

The `<TypePermissions>` root element supports any number of child `<ItemType>` elements that specify the type security permissions. `<ItemType>` is the only permitted child of `<TypePermissions>`:

```
<TypePermissions>
  <ItemType Id="..." SchemaShortName="...">
    ...
  </ItemType>
  ...
</TypePermissions>
```

The `<ItemType>` element has two attributes: `Id`, which is mandatory; and `SchemaShortName`, which is optional:

- `Id` is the identifier of the item type, as defined in the schema that contains it.
- `SchemaShortName` is the short name of the schema that contains the item type. When this attribute is set, it overrides `DefaultSchemaShortName` in the parent element.

Each item type for which the file contains permissions appears in exactly one `<ItemType>` element. If an `<ItemType>` element is empty, however, it is as if that element does not exist.

<Allow>

The item type security model assumes that if you want to control access to a particular type, then usually you want to make it so that only users in particular groups can see records that have that type.

The `<ItemType>` element supports a single `<Allow>` child element. As soon as you add the element, access to the type is denied to all groups that are not specifically mentioned:

```
<TypePermissions>
  <ItemType Id="...">
    <Allow>
      ...
    </Allow>
  </ItemType>
  ...
</TypePermissions>
```

The `<Allow>` element has no attributes. If an `<ItemType>` element has an empty `<Allow>` child element, then only users who have the `i2:Administrator` command access permission can see records of that type.

<UserGroup>

The `<Allow>` element supports any number of child `<UserGroup>` elements. Members of each user group that you specify (as well as users who have the `i2:Administrator` command access permission) are allowed to see records that have the parent item type:

```
<TypePermissions>
  <ItemType Id="...">
    <Allow>
      <UserGroup Name="..." />
      ...
    </Allow>
  </ItemType>
  ...
</TypePermissions>
```

The `<UserGroup>` element has a single, mandatory `Name` attribute. For each user group that should have permission to see records of the specified type, the `<Allow>` element must contain a `<UserGroup>` element whose `Name` attribute is set to the name of the user group.

Changing how item type access is determined

In some deployments of i2 Analyze, it is necessary to configure the system so that users are only able to access and interact with records with particular item types. By default, i2 Analyze determines which item types a user can access by their user group membership using the `type-access-configuration.xml` file. If this suits your needs, see [Item type security](#) to learn how to configure users' access to item types in this way. To determine which item types a user can access based on other user information, you must implement the item type security SPI.

By implementing the item type security SPI, user item type access can be determined by any combination of:

- User principal name
- User group membership
- User security dimension access permissions (defined by the i2 Analyze security schema)

The SPI is a set of Java interfaces that you can implement with your own Java classes. Then, you can configure i2 Analyze to use your implementation instead of the default one.

Implementing the SPI

See the `com.i2group.disco.security.spi` package in the [i2 Analyze Security SPI documentation](#) for details, but the two interfaces that you need to implement are:

- `IUserItemTypeAccessResolver`. This interface is used to get the access levels assigned to a specific user for each item type.
- `IUserItemTypeAccessResolverProvider`. This is the interface that i2 Analyze uses to obtain `IUserItemTypeAccessResolver` instances for specific users. It also exposes a method that allows you to validate and reload the item type security configuration on server startup and reload.

You must package your SPI implementation into a single JAR file.

Configuring i2 Analyze to use your implementation

Note: Customizing item type access resolution should take place during the development phase of your deployment, in the configuration development environment. See [Deployment phases and environments](#) for more information.

For i2 Analyze to use your item type security implementation, you must:

1. Copy the JAR file containing your implementation into the `toolkit/configuration/fragments/opa1-services/WEB-INF/lib` i2 Analyze toolkit directory where it will be deployed into the application.
2. Instruct i2 Analyze to use your implementation by setting the `IUserItemTypeAccessResolverProvider` property in the `toolkit/configuration/fragments/opa1-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file to the fully qualified class name of your `IUserItemTypeAccessResolverProvider` implementation. For example:

```
IUserItemTypeAccessResolverProvider=com.example.ImplementationClassName
```

3. Redeploy and restart liberty. From the `toolkit/scripts` directory, run:

```
setup -t stopLiberty
setup -t deployLiberty
setup -t startLiberty
```

High availability deployments

For high availability and disaster recovery deployment topologies, performing the above procedure in the configuration development environment ensures that your SPI implementation is copied to each Liberty server when you configure the pre-production environment. To make changes to the SPI implementation in a pre-production or test environment that has multiple Liberty servers, you must ensure that the updated JAR file is copied to the toolkit of each Liberty server.

Configuring user security

In a deployment of i2 Analyze, you can configure how users are authenticated and authorized with the system.

Configuring the Liberty user registry

The access levels that each user receives within i2 Analyze are determined by their membership of groups. The names of these groups must match the group permissions elements values that are defined in your security schema.

In the production deployment process, you must configure the Liberty user registry when you modify the groups in the security schema in the schema development or configuration development environments.

The following rules apply:

1. You must create groups in the Liberty user registry whose names exactly match the UserGroup attribute of the group permissions elements in the security schema.
2. You must ensure that every user is a member of enough groups such that they are assigned a dimension value and level from each access security dimension. A user does not require a mapping to a grant security dimension.

To illustrate these rules, consider that the example security schema defines the following dimensions and groups:

Group Permissions UserGroup value	Group Permissions for Dimension	Dimension values and level
Analyst	Security Compartment	Human Informants - update, Open Source Intelligence - read_only
Clerk	Security Compartment	Open Source Intelligence - update
Controlled	Security Level	Controlled - update
Unclassified	Security Level	Controlled - update, Unclassified - update
Security Controller	Grant Access	Security Controller - update

To map to this security schema, the user group values in the table must match with the user groups in the user repository.

Each user in this deployment must be in either of the "Analyst" or "Clerk" groups, and either of the "Controlled" or "Unclassified" groups.

Every deployment must contain an account that is associated with the administrator role. You can create a group in the user registry named "Administrator", or you can change the value of the `security.administrator.group` property to the name of an existing group in the repository. The `security.administrator.group` property is in the `environment-advanced.properties` file for each application, in the `toolkit\configuration\environment\application` directory. When an i2 Analyze user is a member of this group, they can access administrative features.

The following process is an approach to security in Liberty that uses a basic user registry.

1. Create the users and groups in Liberty for each of the group permissions elements in the security schema.
 - a) In an XML editor, open the `user.registry.xml` file. You can find this file in the `C:\IBM\i2analyze\deploy\wlp\usr\shared\config` directory of your Liberty installation.

b) Use the following template to add your users and groups to the `user.registry.xml` file as the first child of the `<server>` element:

```
<basicRegistry id="basic" realm="WebRealm">
  <user name="" password="" />
  <group name="">
    <member name="" />
  </group>
</basicRegistry>
```

Use the following information to populate the template:

- There is a `<user>` element for each user of the system. The `<user>` element's name and password attributes must be populated for that user.
- There is a `<group>` element with a name attribute that matches the name of each security dimension in the security schema.
- The `<group>` elements are populated by `<member>` elements. For a user to be a member of a group, a `<member>` element's name attribute must match that user's name attribute.

If you are using the example deployment, the user Jenny is a member of each group.

In the following example `user.registry.xml`, the users `Analyst1`, and `Clerk1` have been added into a subset of the groups. If you use the following example, log in as these users to see the different permission levels of each group:

```
<basicRegistry id="basic" realm="WebRealm">
  <user name="Jenny" password="{xor}FToxMSY=" />
  <user name="Analyst1" password="{xor}FToxMSY=" />
  <user name="Clerk1" password="{xor}FToxMSY=" />
  <group name="Analyst">
    <member name="Jenny"/>
    <member name="Analyst1"/>
  </group>
  <group name="Clerk">
    <member name="Jenny"/>
    <member name="Clerk1"/>
  </group>
  <group name="Controlled">
    <member name="Jenny"/>
    <member name="Analyst1"/>
  </group>
  <group name="Unclassified">
    <member name="Jenny"/>
    <member name="Clerk1"/>
  </group>
  <group name="Security Controller">
    <member name="Jenny"/>
  </group>
  <group name="Administrator">
    <member name="Jenny"/>
  </group>
</basicRegistry>
```

The passwords can be encoded by the Liberty security utility.

2. Use the Liberty **securityUtility** command to encode the password for each user.

- a) Navigate to the `bin` directory of your WebSphere Application Server Liberty profile deployment that is configured by the deployment toolkit. By default Liberty is deployed in the `C:\IBM\i2analyze\deploy\wlp` directory.
- b) In a command prompt, run the following command:

```
securityUtility encode password
```

The encoded password is displayed in the command line. Record the encoded password, including the `{xor}` prefix, and use the encoded password as the password in the `user.registry.xml` file.

For more information about using the security utility, see [securityUtility command](#).

3. Save and close the file.

To test that your changes have worked, log in to i2 Analyze as one of the users that you added to the user registry.

After you test your changes to the user registry, you can configure user access to features. To access the REST endpoints, a user must be a member of a group that has the `i2:Administrator` permission under command access control. For more information, see [“Configuring command access control”](#) on page 143.

Configuring SPNEGO single sign-on for i2 Analyze

The following section describes how to configure IBM i2 Analyze with Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) single sign-on. The instructions detail how to configure SPNEGO single sign-on with an existing deployment of i2 Analyze.

Intended audience

In the production deployment process, you might first configure SPNEGO single sign-on in the configuration or pre-production environments. As you move to a production deployment, you must replicate any configuration changes in any new deployments.

This section is intended for readers who are familiar with configuring and managing domain controllers, Microsoft Active Directory, and have an understanding of SPNEGO single sign-on.

There are many different single sign-on technologies. This section defines a SPNEGO single sign-on setup with workstations that are members of the same Microsoft Active Directory domain. i2 Analyze uses the users and groups in Active Directory to determine the authorization of users.

The instructions assume that the following prerequisites are installed and accessible:

- A Microsoft Windows® Server running an Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC).
- A Microsoft Windows® domain member (client) with a web browser that supports the SPNEGO authentication mechanism.
- A working deployment of i2 Analyze that can be accessed by users in Active Directory.

For information on the prerequisites that are required, see the [Before you begin](#) section of [Configuring SPNEGO authentication in Liberty](#).



Attention: IBM takes reasonable steps to verify the suitability of i2 Analyze for internet deployment. However, it does not address lower-level issues such as guarding networks

against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. IBM accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

i2 Analyze with SPNEGO single sign-on

In the production deployment process, i2 Analyze is configured to use user names and passwords that are stored in a file-based registry. By configuring the deployment to use SPNEGO single sign-on, a user is logged in through the domain client workstation that they are logged in to.

After a user logs in to a single sign-on environment, they are authenticated with any systems that they have access to. i2 Analyze can be configured to allow authentication through SPNEGO single sign-on, with authorization through Active Directory.

SPNEGO single sign-on enables users to log in to a Microsoft domain controller, and be authenticated within the single sign-on environment. In SPNEGO single sign-on, to change the user that is logged in to i2 Analyze, the user must log out of the workstation, and a new user must log in to the workstation.

SPNEGO single sign-on planning

Planning an implementation of SPNEGO single sign-on with i2 Analyze ensures that the system you create matches the needs of your organization. It is important to understand the organizational requirements and your environment before you begin to plan.

Before you start implementing the solution, ensure that you understand the following aspects of the proposed system:

- What SPNEGO single sign-on is, and the implications of implementing it in your environment. For more information, see [Single sign-on for HTTP requests using SPNEGO web authentication](#).
- The physical architecture that is required to use SPNEGO single sign-on.
- The value of the UserGroup attribute of each <GroupPermissions> element to use with your i2 Analyze deployment. For more information, see the [i2 Analyze Security White Paper](#).

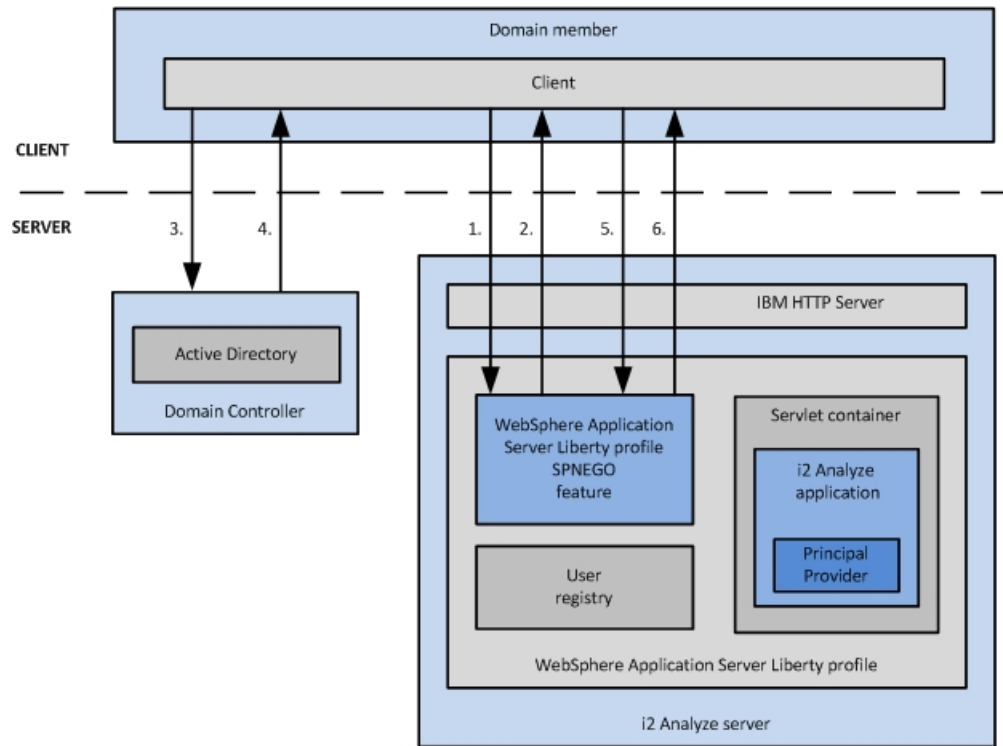
i2 Analyze with SPNEGO single sign-on model

Configuring i2 Analyze to use SPNEGO single sign-on changes the way that users authenticate with the platform. A deployment that uses SPNEGO single sign-on requires the user to access i2 Analyze on a workstation that is a member of the same domain as i2 Analyze.

Authentication

When i2 Analyze is configured to use SPNEGO single sign-on, the authentication sequence between the client and the platform matches the following steps and the associated diagram:

1. The client attempts to connect to WebSphere Application Server Liberty profile with an HTTP/Post/Get request.
2. WebSphere Application Server Liberty profile returns HTTP 401 with a Negotiate header.
3. The client requests a SPNEGO token from the domain controller.
4. The domain controller returns a SPNEGO token to the client.
5. The client attempts to connect to WebSphere Application Server Liberty profile with an HTTP/Post/Get request and the SPNEGO token.
6. On successful authentication, the client receives a Lightweight Third-Party Authentication (LTPA) token in a cookie. During normal operation, the client passes the cookie back to i2 Analyze.

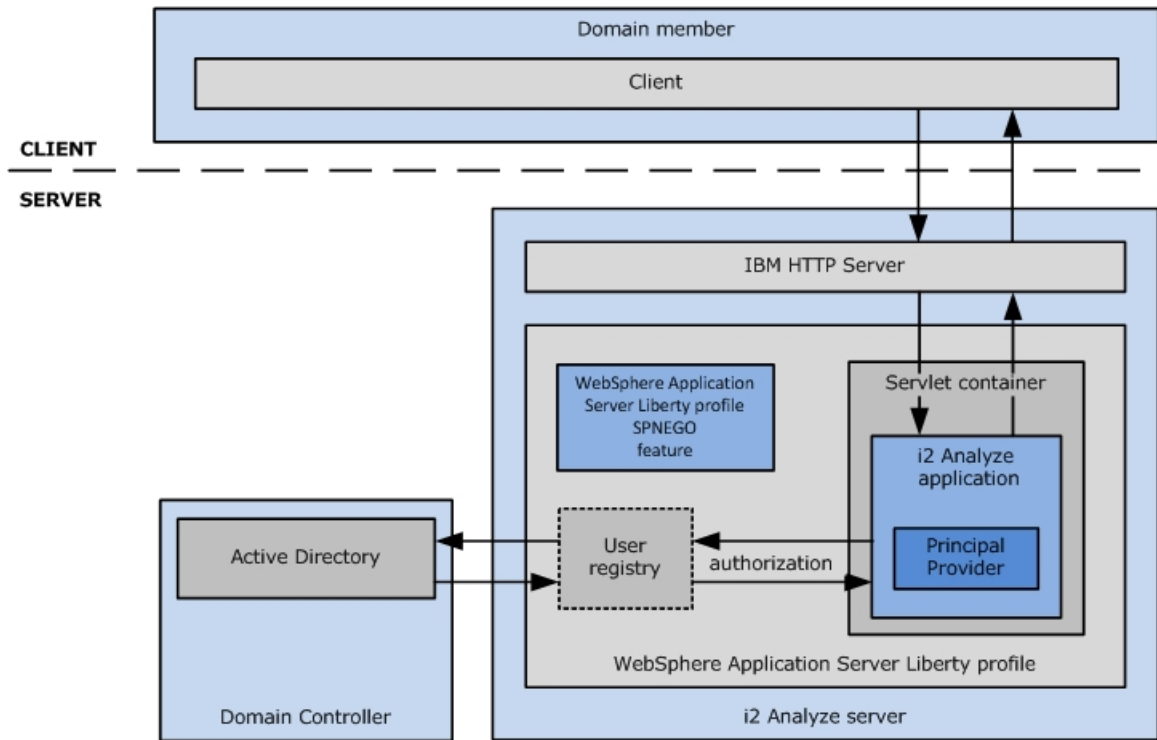


Authorization

After the user is authenticated, they are logged in to i2 Analyze. To define the data that the user has access to, the user must be authorized by i2 Analyze.

For authorization, the i2 Analyze application communicates with Active Directory, through the WebSphere Application Server Liberty profile user registry APIs to retrieve information about the current user. The principal provider then maps the retrieved information to security dimension values in the i2 Analyze security schema.

The following diagram shows how authorization works in i2 Analyze:



Implementing SPNEGO single sign-on

SPNEGO single sign-on enables users to log in to a domain client workstation, and be authenticated with i2 Analyze. Complete any configuration changes to i2 Analyze, and test the system.

- For information on the prerequisites that are required, see the Before you begin section of [Configuring SPNEGO authentication in Liberty](#).
- If you previously deployed i2 Analyze with basic authentication, you must remove or comment out the complete <basicRegistry> element from your user registry.

Populate Active Directory with the correct users and groups for your environment. Then, complete the configuration steps for WebSphere Application Server Liberty profile and the i2 Analyze application. Redeploy i2 Analyze, and connect to i2 Analyze from a client workstation.

Configuring Microsoft Active Directory

The users that are in Microsoft Active Directory are used to authenticate with i2 Analyze. The groups that are in Active Directory are used for authorization in i2 Analyze.

Create Microsoft Active Directory groups that match the value of the UserGroup attribute of each <GroupPermissions> element in the i2 Analyze security schema file.

The groups that you create in Microsoft Active Directory are used for authorization in i2 Analyze. To identify the groups correctly, you must ensure that the names of groups in Active Directory exactly match the value of the UserGroup attribute of each <GroupPermissions> element in the security schema.

Note: The security schema that the deployment uses is defined in the ApolloServerSettingsMandatory.properties file. The security schema, and properties files are in the toolkit\configuration\fragments\common\WEB-INF\classes directory.

In a single sign-on setup, the following users must be present in Active Directory:

- A user for the server that hosts the i2 Analyze application, that is mapped to a Kerberos Service Principal Name (SPN).
- The users that are used to log in to i2 Analyze.

To authorize users, the following groups must be present in Active Directory:

- A group for each of the group permission elements in the i2 Analyze security schema.
- A group for administrators.

1. Create the Microsoft Active Directory groups.

For more information, see [How to Create a Group in Active Directory](#).

- a) Open the Microsoft Active Directory groups controller.
- b) Create groups whose names exactly match the value of the UserGroup attribute of each <GroupPermissions> element in the i2 Analyze security schema file.

2. Create any Microsoft Active Directory users.

Create user accounts that can be used to log in to i2 Analyze.

For more information, see [How to Create a Domain Account in Active Directory](#).

3. Make each user a member of the correct groups for your environment.

The groups that the user is a member of in Active Directory are used for authorization in i2 Analyze.

For more information, see [Adding Users to an Active Directory Group](#).

The users that can access i2 Analyze are created, and are members of the groups that define their access levels.

Configuring WebSphere Application Server Liberty profile

Create the Kerberos service principal name (SPN) and keytab file for the server that hosts WebSphere Application Server Liberty profile that runs the i2 Analyze application. Edit the WebSphere Application Server Liberty profile configuration to use SPNEGO single sign-on and the Active Directory registry.

To use SPNEGO single sign-on with the Liberty server that is deployed by i2 Analyze and other WebSphere Application Server servers, each server must use the same LTPA keys file. For more information about LTPA, see [Authentication - LTPA](#).

The value that is set for the `ltpakeys.password` property in the `credentials.properties` file must match the password that is required to import the keys from the LTPA keys file. If you change the password in the `credentials.properties` file, you must redeploy i2 Analyze for the password change to take effect.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Configure WebSphere Application Server Liberty profile to use SPNEGO single sign-on by using the first two steps in [Configuring SPNEGO authentication in Liberty](#) as a reference.

- a) Create the Kerberos SPN and keytab files on the domain controller.

Note: Ensure that the host file on the Active Directory server uses the full host name, including the domain name, for the i2 Analyze server. Remove any entries that use only the short name for the i2 Analyze server. The value in the host file must match the value that is used for the SPN.

- b) Configure the server that hosts WebSphere Application Server Liberty profile, and WebSphere Application Server Liberty profile.

2. Configure WebSphere Application Server Liberty profile to use the Microsoft Active Directory registry by using the instructions in [Configuring LDAP user registries with Liberty](#) as a reference.

- a) Complete step 1 to add the features to the `i2analyze\deploy\wlp\usr\servers\opal-server\server.xml` file.
- b) Complete step 4 by using the Microsoft Active Directory Server example to populate the `<ldapRegistry>` element.

Note: This information does not cover the configuration of Secure Sockets Layer (SSL) between WebSphere Application Server Liberty profile and Active Directory. Do not include the `<ssl>` and `<keyStore>` elements from the example, in your `server.xml` file.

- c) Ensure that the mapping between Active Directory and the i2 Analyze security schema is correct. Add the following code after the `<ldapRegistry>` element in the `server.xml` file:

```
<federatedRepository>
  <primaryRealm name="">
    <participatingBaseEntry name=""/>
    <groupSecurityNameMapping inputProperty="cn" outputProperty="cn"/>
  </primaryRealm>
</federatedRepository>
```

Populate the empty name attribute values by using the following information:

- The `<primaryRealm>` element's name attribute has the same value as the `realm` attribute of the `<ldapRegistry>` element.
- The `<participatingBaseEntry>` element's name attribute has the same value as the `baseDN` attribute as the `<ldapRegistry>` element.

By default, all requests to access protected resources use SPNEGO authentication. If you previously deployed i2 Analyze with basic authentication, you must ensure that the basic registry is not present in the `user.registry.xml` file.

3. Using an XML editor, either remove or comment out the complete `<basicRegistry>` element in the `i2analyze\deploy\wlp\usr\shared\config\user.registry.xml` file.

Configuring the i2 Analyze application

Set the security administrator group to the value used in Microsoft Active Directory.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Using a text editor, open `toolkit\configuration\environment\opal-server\environment-advanced.properties`. Edit the value of the `security.administrator.group` property to the name of the group in Active Directory to use for administrators. For example, `i2Admins`.

The mapping to the administrator group must be in terms of the Common Name that is defined in the `<participatingBaseEntry>` element of the `<federatedRepository>` element in the `server.xml` file. For more information about the `<federatedRepository>` element, see [“Configuring WebSphere Application Server Liberty profile” on page 132](#).

Redeploying i2 Analyze

Redeploy i2 Analyze to update the application with your configuration changes.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

```
setup -t stopLiberty
```

3. Update the i2 Analyze application:

```
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

5. If you are using the IBM HTTP Server, start, or restart, it.

Testing i2 Analyze with SPNEGO single sign-on on the client workstation

To test the SPNEGO single sign-on setup, connect to i2 Analyze from a domain client workstation. The user that is logged in to the domain client is logged in to i2 Analyze.

- Your web browser must be configured according to step 3 in [Configuring SPNEGO authentication in Liberty](#).
- You must be logged in to the client workstation as one of the users in the domain controller, who is in at least one group per security dimension in the i2 Analyze security schema.

Log in to the client workstation as users with different access levels. For each user, complete the following steps to demonstrate that authorization is working correctly when you are using SPNEGO single sign-on.

1. Open your web browser, and navigate to `http://host_name/opal` (where *host_name* is the fully qualified domain name or IP address of the server that hosts the i2 Analyze application).
2. Create, browse, and search for data, to ensure that you are authenticated and authorized with the platform correctly.

Note: When you create items, ensure that the permissions are set up so that you have access to view them.

After you test your changes, you can configure user access to features. To access the REST endpoints, a user must be a member of a group that has the `i2:Administrator` permission under command access control. For more information, see [“Configuring command access control”](#) on page 143.

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although IBM does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate, and maintain applications that can be trusted. In particular, see https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks and https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

SANS Institute

The System-Admin, Audit, Network, and Security Institute is the largest source for [information security training and security certification](#) in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the internet's early warning system - the [Internet Storm](#)

Center. In particular, see <https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines>

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

In particular, see <http://cwe.mitre.org/top25/index.html#CWE-307>.

Configuring X.509 client certificate authentication with i2 Analyze

You can enable your deployment to use X.509 client certificate authentication. After successful configuration, users can log in to i2 Analyze with client certificates instead of user names and passwords.

After you configure client certificate authentication, a user does not need to enter a user name and password separately. Each certificate is associated with a single user in the user registry to enable authentication. Anyone that has access to a client certificate can log in to i2 Analyze as the user associated with that certificate without entering a password.

To enable a user to log in using a client certificate, the client certificate must be installed in the user's personal certificate store on the workstation they are using to access i2 Analyze. After the client certificate is installed in the personal certificate store, the user can use the certificate to log in to i2 Analyze through Analyst's Notebook Premium.

When a user connects using Analyst's Notebook Premium, the user chooses the certificate to use to log in when they open a connection to the Information Store.

Intended audience

In the production deployment process, you might first configure client certificate authentication in the configuration or pre-production environments. As you move to a production deployment, you must replicate any configuration changes in any new deployments.

This information is intended for readers who are familiar with managing key databases and certificates, user authentication mechanisms, and the i2 Analyze toolkit.

Prerequisites

The starting point for configuring client certificate authentication is a deployment of i2 Analyze that is configured to use Secure Sockets Layer on connections to the HTTP Server, and between the HTTP Server and Liberty. For more information about configuring Secure Sockets Layer on connections to the HTTP Server, see [Configuring Secure Sockets Layer with i2 Analyze](#).



Attention: IBM takes reasonable steps to verify the suitability of i2 Analyze for internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. IBM accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

Client certificates

The client certificates that are used to authenticate users must be signed by a certificate authority that is trusted by the i2 Analyze server.

The common name in a client certificate must match a user name in the i2 Analyze user registry. A user that selects such a certificate logs in to i2 Analyze as the corresponding i2 Analyze user.

You can have as many client certificates as you require. Each certificate is associated with a single user in the user registry. Each certificate can be installed on any number of workstations. Each workstation can have any number of certificates installed.

To demonstrate a working configuration, you can use a self-signed client certificate. For more information, see [“Creating a self-signed client certificate” on page 136](#). However, in a production deployment you must use certificates that are signed by a certificate authority that is trusted by the i2 Analyze server.

There are many methods for obtaining an X.509 certificate that is signed by a certificate authority. When you receive a signed certificate, you also receive signer certificates so that you can trust the client certificates that are signed by that certificate authority. If the certificate authority that signed your certificates is not already trusted within the key database, you must add any signer certificates to the key database so that the certificate authority is trusted.

For information about managing key databases, certificates, and trusted certificate authorities using the IBM Key Management utility, see [Managing keys with the IKEYMAN graphical interface](#).

Creating a self-signed client certificate

The client certificate is used to log in and authenticate a user with i2 Analyze. Use the IBM Key Management utility to create a self-signed certificate.

Create a self-signed certificate to use as a client certificate to demonstrate a working configuration. If you are using client certificates that are signed by a certificate authority, you do not need to complete the following instructions.

1. Start the IBM Key Management utility.

For more information, see [Starting the Key Management utility user interface](#).

Note: The IBM Key Management utility uses a GUI or Window Manager. If you do not have a GUI or Window Manager on your system, you can use the command line interface to complete the same actions. For more information about the command line interface, see [Key Management utility command-line interface \(gskcmd\) syntax](#).

2. Open the key database that is used for Secure Sockets Layer (SSL) connections. If you followed the instructions to set up the SSL example, the key database file is `IBM\i2analyze\i2-http-keystore.kdb`.

For more information about opening a key database, see [Working with key databases](#).

3. Create a self-signed certificate.

For more information, see [Creating a self-signed certificate](#).

- a) Set the **Key Label** to a value that enables you to identify the user.

For example, Jenny.

- b) Ensure that the value of **Common Name** matches the name of a user in the user registry for i2 Analyze.

If you are using the example user registry, set the value of **Common Name** to Jenny.

Note: The user name cannot contain a comma (,).

For this example, you can use the default values for the remaining properties.

4. Export the certificate and private key from the key database.

- a) Click **Export/Import**.
 - b) Ensure that **Export Key** is selected.
 - c) From the **Key file type** list, select PKCS12.
 - d) Set the **File name** to a value that enables you to identify the user.
For example, Jenny.p12.
 - e) Ensure that **Location** is set to the same directory as the key database.
 - f) Click **OK**.
 - g) When you are prompted, provide a password that is used to access the keys.
5. Extract the certificate from the key database.
- a) Click **Extract Certificate**.
 - b) From the **Data type** list, select Binary DER data.
 - c) Set the **Certificate file name** to a value that enables you to identify the user.
For example, Jenny.der.
 - d) Ensure that **Location** is set to the same directory as the key database.
 - e) Click **OK**.

Configuring the key databases

To enable the i2 Analyze server to trust the client certificates, you must ensure that the signer of your client certificates is trusted within the i2 Analyze key database. You must also create a copy of the keystore as a truststore that WebSphere Application Server Liberty uses.

If you are using client certificates that are signed by a certificate authority, ensure that the certificate authority that signed the certificates is trusted within the i2 Analyze key database.

You can list the certificate authorities that are trusted within a key database in the IBM Key Management utility. For more information, see [Listing certificate authorities](#).

After you add the certificate to the key database, create a truststore that WebSphere Application Server Liberty uses. The truststore is a copy of the i2 Analyze key database.

1. Start the IBM Key Management utility.

For more information, see [Starting the Key Management utility user interface](#).

Note: The IBM Key Management utility uses a GUI or Window Manager. If you do not have a GUI or Window Manager on your system, you can use the command line interface to complete the same actions. For more information about the command line interface, see [Key Management utility command-line interface \(gskcmd\) syntax](#).

2. Open the key database that is used for Secure Sockets Layer (SSL) connections. If you followed the instructions to set up the SSL example, the key database file is IBM\i2analyze\i2-http-keystore.kdb.

For more information about opening a key database, see [Working with key databases](#).

3. Add the certificates to the key database, to ensure that the certificates received from the client are trusted.

- a) In the IBM Key Management utility, with the i2 Analyze key database open, select **Signer Certificates** from the list in the **Key database content** pane.
- b) Click **Add**.
- c) Click **Browse**, and locate your certificate.

Note: When you are using a self-signed client certificate, add the self-signed client certificate as a signer certificate. For example, Jenny.der.

4. The Liberty truststore must contain the certificates to ensure that the certificates received from the client are trusted.
 - a) Run the following command to import the required certificate into the truststore. If the truststore does not exist, it is created.:

```
keytool -importcert -alias "signerKey" -keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks" -file "C:\IBM\i2analyze\signer-certificate.cer" -storepass "password"
```

Note: When you are using a self-signed client certificate, add the self-signed client certificate as a signer certificate. For example, Jenny.der.

The key database contains the signer certificates so that the client certificates can be trusted. The truststore is populated so that Liberty can use it to trust the client certificates.

Configuring i2 Analyze

To enable a user to log in using a client certificate, you must modify some of the configuration files for i2 Analyze.

Add a rewrite rule that enables client authentication on the IBM HTTP Server to the i2 Analyze configuration. Then, update the web.xml file for the application to enable client certificate authentication.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using a text editor, open the configuration\environment\proxy\http-custom-rewrite-rules.txt file. Add the following line between the !Start_After_Rules! and !End_After_Rules! lines to enable client certificate authentication:

```
SSLClientAuth Optional
```

2. In an XML editor, open the toolkit\configuration\environment\topology.xml file.

- a) Add a child <key-store> element to the <key-stores> element.

For your truststore, specify the type as trust-store and file as the full path to your truststore. For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
  ...
  <key-stores>
    <key-store type="key-store"
      file="C:\IBM\i2analyze\i2-liberty-keystore.jks"/>
    <key-store type="trust-store"
      file="C:\IBM\i2analyze\i2-liberty-truststore.jks"/>
  </key-stores>
  ...
</application>
```

- b) Specify the truststore password in the credentials file. In a text editor, open the toolkit\configuration\environment\credentials.properties file and enter a password for the truststore that you specified in the topology.xml file.

```
ssl.truststore.password=password
```


3. Using an XML editor, open the `wlp\usr\servers\opal-server\server.xml` file.

- a) Modify the `<ssl>` element with the id `defaultSSLConfig` to include `clientAuthenticationSupported="true"`.

For example:

```
<ssl clientAuthenticationSupported="true"
  id="defaultSSLConfig"
  keyStoreRef="defaultKeyStore"
  trustStoreRef="defaultTrustStore"/>
```

- b) Modify the `<httpDispatcher>` element to include `trustedSensitiveHeaderOrigin="*"`.

For more information about the values that you can provide for the `trustedSensitiveHeaderOrigin` attribute, see [HTTP Dispatcher \(httpDispatcher\)](#).

For example:

```
<httpDispatcher enableWelcomePage="false"
  trustedSensitiveHeaderOrigin="*"/>
```

4. Use an XML editor to modify the `toolkit\configuration\fragments\opal-services-is\WEB-INF\web.xml` file.

Comment out the following lines so that form based authentication is not used:

```
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>Form-Based Authentication Area</realm-name>
  <form-login-config>
    <form-login-page>/login.html</form-login-page>
    <form-error-page>/login.html?failed</form-error-page>
  </form-login-config>
</login-config>
```

In the login configuration section, add the following lines to define the client certificate authentication method:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>WebRealm</realm-name>
</login-config>
```

The i2 Analyze application is configured to allow client certificate authentication.

Redeploying i2 Analyze

Redeploy i2 Analyze to update the application with your configuration changes.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

```
setup -t stopLiberty
```

3. Update the i2 Analyze application:

```
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

5. If you are using the IBM HTTP Server, start, or restart, it.

Installing client certificates

After you create the client certificates, install the client certificates on the client workstations. After the client certificates are installed, they are available to be selected by users to use to log in to i2 Analyze.

On each client workstation that you want a user to be able to log in from, install the client certificate and exported keys as a Personal certificate. You can install multiple certificates and their associated keys on a workstation to allow multiple users to log in.

If you are not using a self-signed certificate, install the signer certificate that is associated with your client certificate so that your operating system can verify the client certificates.

1. Copy your client certificate and the exported keys to a permanent directory on the client workstation.
2. Install the client certificate and exported keys to the **Personal** store:
 - a) Double-click the keys file.
For example, Jenny.p12.
The **Certificate Import Wizard** is displayed.
 - b) Click **Next**.
 - c) Click **Browse**, and locate the keys file to import.
For example, Jenny.p12. Then, click **Next**.
 - d) Enter the password that you specified when the keys were exported from the key database, and click **Next**.
 - e) Click **Place all certificates in the following store**.
 - f) Click **Browse**, and select **Personal**.
 - g) Click **Next**, and then click **Finish**.
3. If you are using a self-signed certificate, install the client certificate to the **Trusted Root Certification Authorities** store:
 - a) Double-click the client certificate file.
For example, Jenny.der.
 - b) Click **Install Certificate**, and then click **Next**.
 - c) Click **Place all certificates in the following store**.
 - d) Click **Browse**, and select **Trusted Root Certification Authorities**.
 - e) Click **Next**, and then click **Finish**.
 - f) If the operating system cannot verify the certificate, a security warning is displayed.
Click **Yes** to accept the certificate.

The client certificate and exported keys are installed. Users can select a certificate to use to log in to i2 Analyze.

Testing the deployment

You can use X.509 certificates to authenticate users in i2 Analyze from Analyst's Notebook Premium. The connection to the i2 Analyze server is secured, and the user is authenticated with the server by the client certificate that is provided by Analyst's Notebook Premium.

- A client certificate must be installed on the client workstation. For more information about installing the client certificates, see [“Installing client certificates” on page 140](#).
- The application server must be running.

Connect to the Information Store, and select a client certificate to use to authenticate with the i2 Analyze server.

1. Enter the name of the repository, and the URL of the server to connect to.

Use the HTTPS protocol to connect to the server. For example, `https://host_name/opa1` (where `host_name` is the fully qualified domain name or IP address of the HTTP server).

You are prompted to provide a certificate.

2. Select the client certificate to use to authenticate with the server from the list that is displayed.

When client certificate authentication is configured correctly, you are logged in to i2 Analyze as the user associated with the selected certificate.

Configuring Liberty to use OCSP to check certificates

After you configure i2 Analyze to use client certificate authentication, you can configure Liberty to use Online Certificate Status Protocol (OCSP) to check the revocation state of the client certificates. For more information about OCSP, see [Online Certificate Status Protocol](#).

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. In a text editor, open the `i2analyze\deploy\wlp\usr\servers\opa1-server\jvm.options` file and add the following lines:

```
-Dcom.sun.net.ssl.checkRevocation=true
-Djava.security.properties=C:/IBM/i2analyze/deploy/wlp/usr/servers/opa1-server/java-security-ocsp.properties
```

Where:

- `checkRevocation` is set to `true` to instruct Liberty to check whether certificates have been revoked.
 - `java.security.properties` is the path to a properties file that contains the settings to configure OCSP.
2. In a text editor, create the `i2analyze\deploy\wlp\usr\servers\opa1-server\java-security-ocsp.properties` file and add the following lines:

```
ocsp.enable=true
ocsp.responderURL=
```

Where:

- `ocsp.enable` is set to `true` to enable OCSP.
- `ocsp.responderURL` is the URL of the OCSP service that is used to check the status of a certificate. When this value is specified, it overrides the value in the Authority Information Access extension on the certificate.

3. Restart Liberty:

```
setup -t restartLiberty
```

Log in to your deployment to test that revoked certificates are identified successfully.

If a user attempts to log in with a revoked certificate, a message is displayed in the Liberty logs. For example:

```
java.security.cert.CertPathValidatorException: Certificate has been revoked, reason: UNSPECIFIED,
  revocation date: Wed Jan 20 17:13:35 UTC 2021, authority: CN=ocsp, OU=i2, O=IBM, ST=England, C=GB, extension
OIDs: []
```

If your OCSP service is unavailable, a message is displayed in the Liberty logs. For example:

```
The extended error message from the SSL handshake exception is: PKIX path validation failed:
java.security.cert.CertPathValidatorException: Unable to determine revocation status due to network error
```

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although IBM does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate, and maintain applications that can be trusted. In particular, see https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks and https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

SANS Institute

The System-Admin, Audit, Network, and Security Institute is the largest source for [information security training and security certification](#) in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the internet's early warning system - the [Internet Storm Center](#). In particular, see <https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines>

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

In particular, see <http://cwe.mitre.org/top25/index.html#CWE-307>.

Controlling access to features

You can control access to the features and types of command that are available to i2 Analyze users. To restrict (and in some cases, to allow) access to features, you need to specify a command access control file.

Note: In the production deployment process, you first configure access to features in the configuration development environment.

You can use a command access control file to specify which permissions to assign to the user groups in the user repository for your deployment. The command access control file contains permissions that apply to i2 Analyze and, where applicable, to clients and client services.

When you create an example deployment, i2 Analyze is configured with the command access control file that the example contains. However, when no file is specified in the configuration, access control is not enabled. All authenticated users can access all features except:

- Exporting search results to a CSV file
- Using the i2 Notebook web client
- Uploading i2 Analyst's Notebook charts to the Chart Store in bulk
- Creating alerts through the REST API
- The administrator permission

After you specify a command access control file, access to all features is controlled by that file. Before they can use a feature, you must give the appropriate permission to a group of users.

When you upgrade a system that has access to specific features enabled, ensure that you check for new features that require new permissions. Without updating your file to add permissions, access to the features are denied to all users. For information about any new permissions, see [Configuration and database changes](#).

Configuring command access control

You can use command access control to determine which commands and features users can access. You can create a command access control file to match the specific needs of your deployment.

In any command access control file that you create, the group names in the file must match the names of user groups in the user registry.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Create a command access control file.
 - a) Navigate to the directory in the deployment toolkit that contains the example security schema: `toolkit\configuration\examples\security-schema`.
 - b) Copy the `example-command-access-control.xml` file to the `configuration\fragments\opal-services\WEB-INF\classes` directory, and rename it to `command-access-control.xml`.
2. Modify the command access control file.
 - a) Open the `command-access-control.xml` file in your XSD-aware XML editor. For more information, see [“Setting up your XSD aware XML editor”](#) on page 304.
The associated XSD file is: `toolkit\scripts\xsd\CommandAccessControl.xsd`.
 - b) Use the [reference](#) information to specify the access control that your system requires.
 - c) Save the completed file.
3. To set the command access control file to be used in the deployment:
 - a) Using a text editor, open the `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties` file.
 - b) Specify your command access control file as the value for the `CommandAccessControlResource` property.
For example:

```
CommandAccessControlResource=command-access-control.xml
```

- c) Save the file.

Redeploy i2 Analyze to update the application with your changes.

4. In a command prompt, navigate to the `toolkit\scripts` directory.

5. Stop Liberty:

```
setup -t stopLiberty
```

6. Update the i2 Analyze application:

```
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

Connect to your deployment and test that members of each user group have the correct access to features. Continue to change the configuration until you are satisfied with the access of each user group.

After you set command access control, you can revert to the default state by ensuring that the `CommandAccessControlResource` property in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties` has no value.

Enabling access to the i2 Notebook web client

The default set of command access control permissions do not provide access to the i2 Notebook web client. If you have a deployment of i2 Analyze that includes the Chart Store, and you want users to be able to use the i2 Notebook web client, you must specify it in a command access control file.

As well as command access control permissions, the i2 Notebook web client requires a secure connection between the user's web browser and the i2 Analyze server. For information on setting up a secure connection to the i2 Notebook web client, see [Securing the connection between clients and the HTTP server](#).

By default, when users log in to i2 Analyze through their web browser, they see the i2 Investigate web client. To change that behavior, you must give specific access to the i2 Notebook web client to some or all of your users.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

To provide access to the i2 Notebook web client to users in some or all user groups, you must add the `i2:Notebook` command access permission for the groups in question.

1. If you have not already done so, follow the instructions in [“Configuring command access control” on page 143](#) to set up a command access control file in your deployment of i2 Analyze.
2. In a text editor, modify the command access control file:
 - a) Open the file, which is conventionally named `command-access-control.xml` and found in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes` directory.
 - b) Add permission to access the i2 Notebook web client to one of the user groups.
For example:

```
<CommandAccessPermissions UserGroup="Analyst">
  <Permission Value="i2:RecordsUpload" />
  <Permission Value="i2:RecordsDelete" />
  <Permission Value="i2:RecordsExport" />
  <Permission Value="i2:ChartsUpload" />
  <Permission Value="i2:ChartsDelete" />
  <Permission Value="i2:Notebook" />
</CommandAccessPermissions>
```

3. Save and close the `command-access-control.xml` file.
4. In a command prompt, navigate to the `toolkit\scripts` directory.
5. Stop Liberty:

```
setup -t stopLiberty
```

6. Update the i2 Analyze application:

```
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

The command access control file

The command access control configuration XML file has the following structure, and can contain the following permissions. Use this reference and example information when you create your own configuration file.

- [“File structure” on page 145](#)
- [“Command access permissions” on page 146](#)

File structure

<CommandAccessControl>

The `<CommandAccessControl>` element is the root of the configuration file.

It contains child `<CommandAccessPermissions>` elements.

<CommandAccessPermissions>

The `<CommandAccessPermissions>` element contains the access permissions for groups of users.

The `UserGroup` attribute defines the user group that the access permissions apply to. The value of the `UserGroup` attribute must match a user group from the user registry. To specify that the permissions apply to all user groups, you can use the `*` wildcard.

It contains one or more child `<Permission>` elements.

<Permission>

The `Value` attribute of the `<Permission>` element defines a permission that members of the user group that is specified in the parent `<CommandAccessPermissions>` element has access to.

For the list of values that you can specify for the `Value` attribute, see [“Command access permissions” on page 146](#).

The following example allows users of all groups to upload records and charts, and members of the "Analyst" user group can delete records and charts too:

```
<tns:CommandAccessControl
  xmlns:tns="http://www.i2group.com/Schemas/2018-01-19/CommandAccessControl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.i2group.com/Schemas/2018-01-19/CommandAccessControl
CommandAccessControl.xsd ">

  <CommandAccessPermissions UserGroup="*">
    <Permission Value="i2:RecordsUpload"/>
    <Permission Value="i2:ChartsUpload"/>
    <Permission Value="i2:ChartsRead"/>
    <Permission Value="i2:Notes"/>
  </CommandAccessPermissions>

  <CommandAccessPermissions UserGroup="Analyst">
    <Permission Value="i2:RecordsDelete"/>
    <Permission Value="i2:ChartsDelete"/>
  </CommandAccessPermissions>

</tns:CommandAccessControl>
```

Command access permissions

- [Record and chart permissions](#) control access to commands for record and chart management.
- [Web client permissions](#) control access to features in the web client.
- [Connector permissions](#) control access to connectors when your deployment includes the i2 Connect gateway.
- [Administrator permissions](#) control access to REST API endpoints, including the admin endpoint.

Record and chart permissions:

i2:RecordsUpload

Members of groups that have this permission can create and modify records and upload them to the Information Store.

Without this permission, users can search for records and add them to charts, but cannot upload changes to records.

i2:RecordsDelete

Members of groups that have this permission can delete records that were originally uploaded through Analyst's Notebook Premium.

Without this permission, users can search for records and add them to charts, but cannot delete records from the Information Store.

i2:RecordsExport

Members of groups that have this permission can export records that are returned in search results to a CSV file.

Without this permission, users cannot export records that are returned in search results to a CSV file.

i2:ChartsUpload

Members of groups that have this permission can create and modify Analyst's Notebook charts and upload them to the Chart Store. Modifying a chart includes deleting versions of a chart, but not deleting the chart itself.

Without this permission, users can save Analyst's Notebook charts locally, but cannot upload new charts and modifications to existing charts.

i2:ChartsBulkUpload

Members of groups that have this permission receive access to the **Upload from Folder** feature in i2 Analyst's Notebook Premium that enables users to upload charts from disk to the Chart Store in bulk.

Note: This permission automatically includes the `i2:ChartsUpload` permission. You do not need to give both permissions to the same user groups.

i2:ChartsDelete

Members of groups that have this permission can delete charts that were originally uploaded through Analyst's Notebook Premium.

Without this permission, users cannot delete charts from the Chart Store.

i2:ChartsRead

Members of groups that have this permission can search for and retrieve charts from the Chart Store.

Without this permission, users cannot search for or retrieve charts.

i2:Notes

Members of groups that have this permission can create and access notes on records and charts.

Without this permission, notes are not displayed in the Notes tab, and the contents of any notes are not searchable.

Web client permission:

i2:Notebook

Members of groups that have this permission can access the i2 Notebook web client. Members of groups without this permission see the i2 Investigate web client instead.

For more information, see [“Enabling access to the i2 Notebook web client”](#) on page 144.

Connector permissions:

i2:Connectors

If you are using the i2 Connect gateway, members of groups that have this permission can view all i2 Connect connectors.

Without this permission, i2 Connect connectors are not visible unless individual connectors are specified by using the `i2:Connectors:connector-id` permission.

i2:Connectors:connector-id

If you are using the i2 Connect gateway, members of groups that have this permission can view the i2 Connect connector with the matching `connector-id`. For example, `i2:Connectors:example-connector`.

Without this permission, the specified i2 Connect connector is not visible.

Administrator permissions:

i2:AlertsCreate

Members of groups that have this permission can access the REST API alerts endpoint to create and send alerts to i2 Analyze users. For more information, see [“Managing data in i2 Analyze” on page 367](#).

i2:Administrator

Members of groups that have this permission can access the REST API admin endpoints. For more information, see [“Using the admin endpoints” on page 305](#).

Configuring system limits and variables

The i2 Analyze deployment toolkit contains settings that specify system limits and variables. If you modify these settings in a deployed system, redeploy i2 Analyze to update your deployment with any changes.

Modifying the database configuration

You can configure the Information Store database that is deployed as part of i2 Analyze. The types of configuration that you can modify include certain properties of the database and the indexes that are created.

Information Store database properties

The Information Store database properties files specify the names and details of database objects that are using in the Information Store. There is a separate file for IBM Db2 and Microsoft SQL Server.

The `InfoStoreNamesDb2.properties` and `InfoStoreNamesSQLServer.properties` files are in the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes` directory.

The settings in these files control the following database objects:

- The names of the database schemas
- The name of the deletion by rule role
- The collation sequence
- For Db2, the page size
- For Db2, the names of the buffer pools
- For SQL Server, the logical names and locations of the filegroup files

Note: If you change the values for any of the settings after the Information Store is created, you must either manually change the names of the database objects in your database management system or re-create the database.

Settings for Db2 and SQL Server

The `InfoStoreNamesDb2.properties` and `InfoStoreNamesSQLServer.properties` files contain the following settings:

MetadataSchema

The metadata schema contains objects that relate to the definitions of data structures in the Information Store.

By default, the value is `IS_Meta`

DataSchema

The data schema contains objects that store all the data that is available for analysis.

By default, the value is `IS_Data`

WebChartSchema

The web chart schema contains temporary objects used during manipulation of the web chart.

By default, the value is IS_WC

VisualQuerySchema

The visual query schema contains temporary objects used during visual query processing.

By default, the value is IS_Vq

FindPathSchema

The find path schema contains objects that support find path results.

By default, the value is IS_FP

StagingSchema

The staging schema contains temporary objects that support the ingestion process.

By default, the value is IS_Staging

PublicSchema

The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.

By default, the value is IS_Public

DeletionByRuleRoleName

The deletion-by-rule role name.

By default, the value is Deletion_By_Rule

Collation

The collation sequence used for the Information Store. You can only change this setting before you create the Information Store database. Defaults to CLDR181_LEN_S1 for Db2 and Latin1_General_100_CI_AI_SC for SQL Server.

Settings for Db2

The InfoStoreNamesDb2.properties file contains the following settings that are specific to deployments that use IBM Db2 to host the Information Store database:

SystemTempTableSpace

The system temporary table space.

By default, the value is IS_TEMP_TS

UserTemp16KTableSpace

The user temporary 16K table space to hold global temporary objects.

By default, the value is IS_16K_TS

BigTableSpace

The partitioned table spaces to hold data objects and indexes.

By default, the value is IS_BIG_TS

BigIndexTableSpace

By default, the value is IS_BIG_INDEX_TS

SmallTableSpace

The non-partitioned table spaces to hold data objects and indexes.

By default, the value is IS_SMALL_TS

SmallIndexTableSpace

By default, the value is IS_SMALL_INDEX_TS

LobTableSpace

The table spaces to hold LOB data objects and indexes.

By default, the value is IS_LOB_TS

LobIndexTableSpace

By default, the value is IS_LOB_INDEX_TS

IngestionTempTableSpace

The temporary table space to hold global temporary objects during ingestion.

By default, the value is IS_16K_TS

PageSize

The page size.

By default, the value is 16K

BufferPool16K

The 16K buffer pool.

By default, the value is IS_16K_BP

Settings for SQL Server

The InfoStoreNamesSQLServer.properties file contains the following settings that are specific to deployments that use Microsoft SQL Server to host the Information Store database.

For the Filename settings, you must specify a file name or absolute path. If you specify a file name, it is located relative to the value that is specified in the environment.properties file for the db.database.location.dir property. If you specify a file path, it is considered an absolute path.

ISSchema

The Information Store schema contains internal configuration information about the Information Store database.

By default, the value is IS_Core

PrimaryName

The SQL Server database primary data file contains the startup and configuration information for the Information Store database. The logical name of the primary data file.

By default, the value is IStore_System_Data

PrimaryFilename

The filename or absolute path for the primary data file.

By default, the value is IStore-p1.mdf

UserTableName

The user table filegroup contains all the tables and data for the Information Store database. The logical name of the user table in the user table filegroup.

By default, the value is User_Table_Data

UserTableFilename

The filename or absolute path for the user table file.

By default, the value is IStore-ut1.ndf

IndexName

The index filegroup contains the indexes for the Information Store database. The logical name of the index file.

By default, the value is IStore_Indexes

IndexFilename

The filename or absolute path for the index file.

By default, the value is IStore-i1.ndf

LobName

The LOB filegroup contains the tables that Store LOB data. The logical name of the lob file.

By default, the value is IStore_Lobs

LobFilename

The filename or absolute path for the index file.

By default, the value is IStore-b1.ndf

MemoryOptimizedName

The memory-optimized filegroup holds one or more containers that contain data files, delta files, or both for memory-optimized tables. The logical name of the memory-optimized file.

By default, the value is IStore_Memory_Optimized

MemoryOptimizedFilename

The filename or absolute path for the memory-optimized file.

By default, the value is IStore-mo1.ndf

SqlLogName1

The transaction log files used for the Information Store. The logical name of the first SQL log.

By default, the value is IStore_Logs_1

SqlLogFilename1

The filename or absolute path for the first SQL log file.

By default, the value is IStore-log1.ldf

SqlLogName2

The logical name of the second SQL log.

By default, the value is `ISStore_Logs_2`

SqlLogFilename2

The filename or absolute path for the second SQL log file.

By default, the value is `ISStore-log2.ldf`

DDLBatchSize

The number of DDL statements in each batch.

By default, the value is 5

DMLBatchSize

The number of DML statements in each batch.

By default, the value is 1000

After you update the database properties, you can either modify other aspects of the deployment toolkit or deploy the system to update the deployment with any changes.

The `InfoStoreNamesDb2.properties` file

The following properties are in the `InfoStoreNamesDb2.properties` file:

MetadataSchema

The metadata schema contains objects that relate to the definitions of data structures in the Information Store.

By default, the value is `IS_Meta`

DataSchema

The data schema contains objects that store all the data that is available for analysis.

By default, the value is `IS_Data`

WebChartSchema

The web chart schema contains temporary objects used during manipulation of the web chart.

By default, the value is `IS_WC`

VisualQuerySchema

The visual query schema contains temporary objects used during visual query processing.

By default, the value is `IS_Vq`

FindPathSchema

The find path schema contains objects that support find path results.

By default, the value is `IS_FP`

StagingSchema

The staging schema contains temporary objects that support the ingestion process.

By default, the value is `IS_Staging`

PublicSchema

The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.

By default, the value is IS_Public

SystemTempTableSpace

The system temporary table space.

By default, the value is IS_TEMP_TS

UserTemp16KTableSpace

The user temporary 16K table space to hold global temporary objects.

By default, the value is IS_16K_TS

BigTableSpace

The partitioned table spaces to hold data objects and indexes.

By default, the value is IS_BIG_TS

BigIndexTableSpace

By default, the value is IS_BIG_INDEX_TS

SmallTableSpace

The non-partitioned table spaces to hold data objects and indexes.

By default, the value is IS_SMALL_TS

SmallIndexTableSpace

By default, the value is IS_SMALL_INDEX_TS

LobTableSpace

The table spaces to hold LOB data objects and indexes.

By default, the value is IS_LOB_TS

LobIndexTableSpace

By default, the value is IS_LOB_INDEX_TS

IngestionTempTableSpace

The temporary table space to hold global temporary objects during ingestion.

By default, the value is IS_16K_TS

InternalStagingTableSpace

The table space to hold internal staging data objects created during ingestion Note: these tables have data inserted using NOT LOGGED INITIALLY

By default, the value is IS_INTSTG_TS

PageSize

The page size.

By default, the value is 16K

BufferPool16K

The 16K buffer pool.

By default, the value is IS_16K_BP

DeletionByRuleRoleName

The deletion-by-rule role name.

By default, the value is Deletion_By_Rule

Collation

The collation sequence used for the Information Store in Db2. You can only change this setting before you create the Information Store database. Defaults to "CLDR181_LEN_S1".

By default, the value is CLDR181_LEN_S1

LoggerSchema

The following options are not used for new Information Store deployments, but might be used during upgrade from earlier versions. You must not modify these settings.

By default, the value is LOGGER

LoggerDataSchema

By default, the value is LOGDATA

LoggerModuleSchema

By default, the value is LOGGER_1B

LoggerTableSpace

By default, the value is LOG_DATA_SPACE

LoggerMetadataTableSpace

By default, the value is LOGGER_SPACE

ReportsSchema

By default, the value is IS_Rpt

The InfoStoreNamesSQLServer.properties file

The following properties are in the InfoStoreNamesSQLServer.properties file:

ISSchema

The Information Store schema contains internal configuration information about the Information Store database.

By default, the value is IS_Core

MetadataSchema

The metadata schema contains objects that relate to the definitions of data structures in the Information Store.

By default, the value is IS_Meta

DataSchema

The data schema contains objects that store all the data that is available for analysis.

By default, the value is IS_Data

WebChartSchema

The web chart schema contains temporary objects used during manipulation of the web chart.

By default, the value is IS_WC

VisualQuerySchema

The visual query schema contains temporary objects used during visual query processing.

By default, the value is IS_Vq

FindPathSchema

The find path schema contains objects that support find path results.

By default, the value is IS_FP

StagingSchema

The staging schema contains temporary objects that support the ingestion process.

By default, the value is IS_Staging

PublicSchema

The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.

By default, the value is IS_Public

DeletionByRuleRoleName

The deletion-by-rule role name.

By default, the value is Deletion_By_Rule

Collation

The collation sequence used for the Information Store in SQL Server. You can only change this setting before you create the Information Store database. Defaults to Latin1_General_100_CI_AI_SC.

By default, the value is Latin1_General_100_CI_AI_SC

PrimaryName

The SQL Server database primary data file contains the startup and configuration information for the Information Store database. The logical name of the primary data file.

By default, the value is IStore_System_Data

PrimaryFilename

The filename or absolute path for the primary data file.

By default, the value is IStore-p1.mdf

UserTableName

The user table filegroup contains all the tables and data for the Information Store database. The logical name of the user table in the user table filegroup.

By default, the value is User_Table_Data

UserTableFilename

The filename or absolute path for the user table file.

By default, the value is IStore-ut1.ndf

IndexName

The index filegroup contains the indexes for the Information Store database. The logical name of the index file.

By default, the value is IStore_Indexes

IndexFilename

The filename or absolute path for the index file.

By default, the value is IStore-i1.ndf

LobName

The LOB filegroup contains the tables that Store LOB data. The logical name of the lob file.

By default, the value is IStore_Lobs

LobFilename

The filename or absolute path for the index file.

By default, the value is IStore-b1.ndf

MemoryOptimizedName

The memory-optimized filegroup holds one or more containers that contain data files, delta files, or both for memory-optimized tables. The logical name of the memory-optimized file.

By default, the value is IStore_Memory_Optimized

MemoryOptimizedFilename

The filename or absolute path for the memory-optimized file.

By default, the value is IStore-mo1.ndf

SqlLogName1

The transaction log files used for the Information Store. The logical name of the first SQL log.

By default, the value is IStore_Logs_1

SqlLogFilename1

The filename or absolute path for the first SQL log file.

By default, the value is IStore-log1.ldf

SqlLogName2

The logical name of the second SQL log.

By default, the value is IStore_Logs_2

SqlLogFilename2

The filename or absolute path for the second SQL log file.

By default, the value is IStore-log2.ldf

ReportsSchema

The following options are not used for new Information Store deployments, but might be used during upgrade from earlier versions. You must not modify these settings.

By default, the value is IS_Rpt

Creating indexes in the Information Store database

In the Information Store, you can add indexes to the item type tables, and to the property type columns of those tables.

Indexes can improve the performance of Visual Query searches. Indexes can also improve the performance of the merged property values definition views if you are using them.

To create an index, you must use the `informationStoreModifications.sql` file. This script is run every time the Information Store database is created. You can also use the `modifyInformationStoreDatabase` toolkit task to run the script at any time.

The file must be in the `configuration\environment\opal-server\databases\infostore` directory. `infostore` is the value of the `id` attribute of the database element for your Information Store database in the `topology.xml` file.

In the Information Store database, the `IS_DATA` schema contains the tables for each item type. Entity item type tables are prefixed with `E_` and link item type tables with `L_`. In these tables, property columns are prefixed with `P_`.

For example, your `i2 Analyze` schema might contain a `Person` entity with a `First Given Name` property. If you know that this property is used a lot by analysts in Visual Query searches, you might want to create an index on that table and column. To create a simple index for the `First Given Name` property, add the following statement to the `informationStoreModifications.sql` file:

```
CREATE INDEX IS_DATA.E_PERSON_FN_IX ON IS_DATA.E_PERSON (P_FIRST_GIVEN_NAME);
```

`IS_DATA.E_PERSON_FN_IX` is the name of the index to create, `IS_DATA.E_PERSON` is the table for the `Person` entity type, and `P_FIRST_GIVEN_NAME` is the column for the first given name property type.

To determine the syntax of the SQL statement that you must use to create the index, use the documentation for your database management system. For more information about creating indexes in a Db2 database, see [CREATE INDEX statement](#) and for SQL Server, see [CREATE INDEX \(Transact-SQL\)](#).

1. Identify the item types, and any of their property types, that you want to add indexes for.
2. Create the directory for the `informationStoreModifications.sql` file.
 - a) In the `configuration\environment\opal-server` directory, create the `databases` directory.
 - b) In the `databases` directory that you created, create the `infostore` directory.

You can find the value to use for `infostore` in your `topology.xml` file. The value to use in your deployment is the value of the `id` attribute of the `<database>` element for your Information Store database.

For example, `configuration\environment\opal-server\databases\infostore`.
3. Using a text editor, create a file that is named `informationStoreModifications.sql` in the `configuration\environment\opal-server\databases\infostore` directory.
4. Develop a script to create the indexes on the tables and columns that you identified in step 1 in the `informationStoreModifications.sql` file.
5. Save and close the file.
6. If the Information Store database exists, run the `modifyInformationStoreDatabase` toolkit task to run the script that you saved in step 4.
 - a) Open a command prompt and navigate to the `toolkit\scripts` directory.
 - b) Run the following command to run the `informationStoreModifications.sql` file:

```
setup -t modifyInformationStoreDatabase
```

Ensure that the indexes you expect are in the Information Store database. You can use IBM Data Studio or SQL Server Management Studio to see the indexes in the Information Store database.

Modifying the remote Db2 database configuration

You can change the configuration attributes of your remote Db2 databases in the i2 Analyze deployment toolkit. To modify the remote database configuration, you must recatalog the remote nodes.

When you recatalog a remote node, the existing node is removed from the Db2 node directory and is then cataloged again with the updated configuration values.

If you need to configure security settings on the connection, you can recatalog the remote nodes manually, instead of using the tasks that are in the deployment toolkit. To recatalog the remote nodes manually, you can use the Db2 **UNCATALOG NODE** and **CATALOG TCP/IP/TCPIP4/TCPIP6 NODE** commands.

You run the **UNCATALOG NODE** and **CATALOG NODE** commands manually instead of using the `recatalogRemoteDB2Nodes` task. For more information about the commands, see [UNCATALOG NODE command](#) and [CATALOG TCP/IP/TCPIP4/TCPIP6 NODE command](#).

Note: The `recatalogRemoteDB2Nodes` and `recatalogRemoteDB2Nodes` tasks use the Db2 **CATALOG TCP/IP NODE** command. The following table shows how the **CATALOG** command parameters map to the values in the `topology.xml` file:

CATALOG TCP/IP NODE command parameters	<database> element attributes
TCPIP NODE <i>nodename</i>	node-name
REMOTE <i>hostname</i>	host-name
SERVER <i>port number</i>	port-number
REMOTE_INSTANCE <i>instance-name</i>	instance-name

1. Change the host name, port, and operating system type attributes of the remote database:
 - a) Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
 - b) Update the `host-name` and `port-number` attribute values of the `<database>` element for the database.
 - c) Run the following command to recatalog the remote node:

```
setup -t recatalogRemoteDB2Nodes
```

2. Change the node name attribute of the remote database:
 - a) Run the following command to uncatalog the remote node:

```
setup -t uncatalogRemoteDB2Nodes
```

- b) Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
- c) Update the `node-name` attribute values of the `<database>` element for the database.
- d) Run the following command to catalog the remote node:

```
setup -t catalogRemoteDB2Nodes
```

The next time that you deploy i2 Analyze, or recreate the database, the database is created using the new values that you provided in the `topology.xml`.

Modifying the maximum number of notes in a record

In a deployment of i2 Analyze, analysts can add notes that contain information that is not categorized by the type of entity or link to Information Store records. The number of notes that a record contains can impact the performance of searching for and uploading that record.

You can configure the maximum number of notes that a record can contain to a number that better matches system requirements. By default, the maximum number of notes that a record can contain is 50.

If a record already contains more notes than the maximum that you set, analysts can continue to edit and delete the existing notes. However, analysts can only create notes in the record by deleting existing notes until the number of notes is less than the new maximum.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Change the value of the `RecordMaxNotes` property.
For example, `RecordMaxNotes=25`.
3. Save your changes.

After you modify the maximum number of notes a record can contain, redeploy the i2 Analyze application for the changes to take effect. For more information about redeploying your system, see [“Redeploying Liberty” on page 304](#).

After you redeploy i2 Analyze, test that the system continues to meet your requirements.

Configuring search

Depending on the requirements of the environment that you are deploying into, a number of configurable options are available for searching and indexing. You can configure the behavior of search features to match your requirements.

Configuring Quick Search

Analysts can use Quick Search to find records in the Information Store by specifying terms that records might contain. You can configure the behavior of Quick Search to meet the language and structure of your data and the requirements of your analysts.

Configuring the Solr index

By default, the Solr index assumes that text data in i2 Analyze follows the conventions of Western European languages, and uses a list of synonyms that contains US English terms. It also treats letters like `o`, `ó`, and `ö` as equivalent for the purposes of filtering search results. You can change these settings so that the index meets the language requirements of your data.

To configure the Solr index, modify the `schema-template-definition.xml` file. In the template definition, you can control the following aspects of Solr:

- How Solr returns data based on the language of the data in the index
- The Solr synonyms file
- Whether to preserve diacritics in search result filters. For example to treat `a`, `å`, and `ä` as equivalent or not.

By default, the Solr search index is configured for Western European languages, and uses synonym lists that contain US English terms. The template definition configuration file includes template options for ar_EG (Arabic) and he_IL(Hebrew) languages.

Each language template has a default synonyms file that is associated with the template language. You can change the default synonyms file to use a customized synonyms file. For more information, see [“Creating a synonyms file”](#) on page 163.

By default, diacritics are not preserved in filters. This means that "Amélie" and "Amelie" both contribute to the "amelie" filter. To ensure that "Amélie" and "Amelie" create separate filters, you can preserve the diacritics.

Note: Changing the diacritic behavior for filters does not impact the behavior when searching.

If you need to configure the behavior of the Solr index, do this in your configuration development environment. If your system contains data, you must reindex after changing the Solr configuration.

1. To modify the Solr index configuration, open the configuration\solr\schema-template-definition.xml file in an XML editor.
2. Specify the language template and synonyms file for the index.
 - a) To specify the language template and default synonyms file, uncomment the section of the template file for the language you want to use.
For example, to use Arabic uncomment the <Definition> and <SynonymsFile> file elements in the ar_EG config section:

```
<!-- ar_EG config -->
<Definition Analyzer="free_text">
  <AnalyzerChain>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.ASCIIFoldingFilterFactory" preserveOriginal="false"/>
  </AnalyzerChain>
  <PostSynonyms>
    <filter class="solr.ArabicNormalizationFilterFactory"/>
    <filter class="solr.ArabicStemFilterFactory"/>
  </PostSynonyms>
</Definition>

<Definition Analyzer="text_facet">
  <AnalyzerChain>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.ASCIIFoldingFilterFactory" preserveOriginal="false"/>
    <filter class="solr.ArabicNormalizationFilterFactory"/>
    <filter class="solr.ArabicStemFilterFactory"/>
  </AnalyzerChain>
</Definition>

<SynonymsFile Path="synonyms-ar_EG.txt" />
```

- b) To specify a different synonyms file, place the file in the configuration\solr directory and provide the file name in the <SynonymsFile> element.
For example:

```
<SynonymsFile Path="custom-synonyms.txt" />
```

For more information about creating a custom synonyms file, see [“Creating a synonyms file”](#) on page 163.

3. To preserve the diacritics in facets, you must complete the previous step for your chosen template and remove the following line from the `text_facet` analyzer:

```
<filter class="solr.ASCIIFoldingFilterFactory" preserveOriginal="false"/>
```

Update the application with your configuration changes.

Note: This procedure removes the data from your Solr index. When Liberty is started, Solr reindexes the data in the Information Store database.

4. On the Liberty server, open a command prompt and go to the `toolkit\scripts` directory.
5. Stop i2 Analyze.
 - a) If you are using a single server deployment, run `setup -t stop`.
 - b) If you are using a multiple server deployment, complete the steps to stop the components of i2 Analyze in [“Stopping and starting i2 Analyze”](#) on page 303.
6. Redeploy i2 Analyze:

```
setup -t deployLiberty
```

7. Create and upload the Solr configuration to ZooKeeper:

```
setup -t createAndUploadSolrConfig --hostname 'liberty.host-name'
```

Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

8. Clear the search index:

```
setup -t clearSearchIndex --hostname 'liberty.host-name'
```

Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

9. Start i2 Analyze
 - a) If you are using a single server deployment, run `setup -t start`.
 - b) If you are using a multiple server deployment, complete the steps to start the components of i2 Analyze in [“Stopping and starting i2 Analyze”](#) on page 303.

Run a selection of queries against your deployment server to test the configuration.

The Solr template configuration file

The Solr template configuration file that is provided in the i2 Analyze deployment toolkit contains commented-out templates for a number of languages.

Root element

- [<SolrSchemaTemplate>](#)

i2 Analyze elements

- [<Definition>](#)
- [<AnalyzerChain>](#)

- [<PostSynonyms>](#)
- [<SynonymsFile>](#)

Solr elements

- [<tokenizer>](#) and [<filter>](#)

Root element

<SolrSchemaTemplate>

<SolrSchemaTemplate> is the root element of the Solr template configuration file. It references the SolrSchemaTemplate.xsd file and version number. Do not change the value for the Version attribute.

```
<SolrSchemaTemplate
  xmlns:tns="http://www.i2group.com/Schemas/2021-02-12/SolrSchemaTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.i2group.com/Schemas/2021-02-12/SolrSchemaTemplate
  SolrSchemaTemplate.xsd "
  Version="1">

</SolrSchemaTemplate>
```

i2 Analyze elements

<Definition>

In the Solr index, analyzers are used to examine the text that is stored in the index. Two of the analyzers that i2 Analyze uses are the free_text and text_facet analyzers. In the template file, the analyzer is specified in the Analyzer attribute of the <Definition> element.

```
<Definition Analyzer="free_text">
...
</Definition>

<Definition Analyzer="text_facet">
...
</Definition>
```

<AnalyzerChain>

The <AnalyzerChain> element is a container for the [“Solr elements” on page 163](#) in the analyzer chain. The <AnalyzerChain> element is a child of the <Definition> element.

```
<Definition Analyzer="free_text">
  <AnalyzerChain>
    ...
  </AnalyzerChain>
</Definition>
```


<PostSynonyms>

The <PostSynonyms> element is a container for the “Solr elements” on page 163 in the analyzer chain that are applied after the synonym filter. The <PostSynonyms> element is a child of the <Definition> element, specified after the <AnalyzerChain> element.

```
<Definition Analyzer="free_text">
  <AnalyzerChain> ... </AnalyzerChain>
  <PostSynonyms>
    ...
  </PostSynonyms>
</Definition>
```

<SynonymsFile>

The Path attribute of the <SynonymsFile> element contains the file name of the synonyms file to use. The <SynonymsFile> element is at the same level as the <Definition> element.

```
<SynonymsFile Path="synonyms-ar_EG.txt" />
```

Solr elements

The <tokenizer> and <filter> are directly converted to Solr. For more information about the elements, see [Understanding Analyzers, Tokenizers, and Filters](#).

In the Solr template configuration, the <tokenizer> and <filter> elements can be child elements of the <AnalyzerChain> and <PostSynonyms> elements.

Note: The <tokenizer> element can be specified as a child of the <Definition> element where Analyzer="free_text".

Creating a synonyms file

Solr provides the facility to configure the synonyms that are used for querying textual data. In i2 Analyze, you can use this option to apply a customized list of synonyms at query time. Synonyms, if not accounted for, can cause a reduction in the relevance of a search result when you search for keywords that are present in alternative forms in your index.

The synonyms file is the part of the Solr configuration that accounts for the presence of synonyms in your data. For example, your data might contain the words, “bag, handbag, pocketbook, purse” for the concept “bag”. When someone searches they are likely to search for one, but expect results for all four. To meet that expectation, you might want to create a customized synonyms file to accommodate similar variations that are specific to your data. The exact words in a synonyms list that are most useful in your deployment depend on the content of your data. You can also use a mix of languages, which might be useful in some contexts, for example names: 'George, Γεώργιος, Jorge'.

The default synonyms file and synonyms list are in US English. The synonyms files that are associated by default with each supported language are supplied in the directory, `toolkit\configuration\solr`.

To customize the alternative terms that are used in search operations for your data, you can create files that contain different terms from those terms that are contained in the supplied synonyms files.

The customized file must adhere to the following guidelines:

- The file must be UTF-8 encoded.
- The terms in the file must match the terms that are produced by the analyzer chain that is used in Solr prior to the synonym filter being applied.

- If multiple forms of a word exist, all the forms must be specified in order for synonym matching to work on each form.
 - Words from Latin script languages, for example French or Italian, must be specified without diacritics. For example, use the following substitution:
 - a instead of á
 - c instead of ç
 - Arabic and Hebrew words must be specified exactly as they are written.
1. Create a text file that defines synonyms in the required Solr format.

For more information, see https://lucene.apache.org/core/8_2_0/analyzers-common/org/apache/lucene/analysis/synonym/SolrSynonymParser.html.

Note:

- a. You cannot search for multi-word terms. However, if you have data that contains terms "USA" and "United States of America", you can search for "USA" and use a synonym to ensure a match with "United States of America".
 - b. You can provide synonyms for terms that include punctuation. However, a search on such a term might not work correctly. The unexpected result is because a filter is applied before synonyms, which means, for example, "Mary-Ann" becomes "Mary,Ann" and then synonyms are expanded from "Mary" and "Ann"; not "Mary-Ann" or "Maryann".
2. Save the file with a .txt extension, for example custom-synonyms.txt.
 3. Complete the instructions in [“Configuring the Solr index”](#) on page 159 to deploy with your synonyms file.

Modifying the wildcard minimum character limits

Wildcard characters are used to match zero or more alphanumeric characters in a search term. You can configure how users complete Quick Search and Visual Queries by modifying the minimum number of characters that must be included in a search term that includes wildcards.

The following wildcard characters are available in i2 Analyze:

*

Matches zero or more alphanumeric characters in this position.
For example, the search term Tim* matches Tim, Time, and Timely.

?

Matches one alphanumeric character in this position.
For example, the search term Tim? matches Time.

Wildcard characters can be used at any position in a search term.

Search terms with wildcard queries might result in a large number of matches, which might cause performance problems. To reduce the possible matches from a wildcard search, you can ensure that users provide a minimum number of characters with a wildcard. For example, the term "*" matches everything. If users must provide a minimum of 3 characters with an asterisk, for example the term "abc*", the number of matches is reduced to values that begin with "abc".

If the minimum number of characters is set too high, users might not be able to search for the terms that they need. For example, in a deployment where the wildcard minimum characters are configured as follows:

- A minimum of 3 characters other than asterisks (*) must be provided in a term.

- A minimum of 2 characters other than question marks (?) and asterisks (*) must be provided in a term.

If the user knows only 2 characters of a license plate, they might not be able to use a wildcard search:

- If the user knows the position of the 2 characters, they can use the question mark wildcard character in the search. If the 2 characters are in positions 2 and 3, then the query "?AB*" is valid in the configuration that is described.
- If the position of the characters is not known, the user might want to search for "*AB*", which is invalid in the configuration that is described.

In addition to wildcard characters that are specifically entered as part of Visual Query, several conditions provide implicit wildcard logic:

- 'Starts with' - Applies an asterisk to the end of the condition. For example, 'Starts with: Fred' is equivalent to 'Fred*', which could match; Fred, Frederick, or Freddie.
- 'Ends with' - Applies an asterisk to the start of the condition. For example, 'Ends with: Fred' is equivalent to '*Fred', which could match; Fred, Wilfred, or Alfred.
- 'Contains' - Applies an asterisk to both the start and the end of the condition. For example, 'Contains: Fred' is equivalent to '*Fred*', which could match any of the above terms, but also include Alfredo.

The use of these conditions follow the same limits as wildcard characters that have been entered explicitly.

To change the minimum number of characters that must be included in a search query with a wildcard character, edit properties in `DiscoServerSettingsCommon.properties`.

The properties that specify the minimum number of characters for Quick Search are:

WildcardMinCharsWithAsterisk

The minimum number of characters other than asterisks (*) that must be included in a wildcard query that contains an asterisk.

WildcardMinCharsWithQuestionMark

The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `WildcardMinCharsWithAsterisk` property.

The properties that specify the minimum number of characters for Visual Query are:

VisualQueryWildcardMinCharsWithAsterisk

The minimum number of characters other than asterisks (*) that must be included in a Visual Query condition that contains or implies asterisks.

VisualQueryWildcardMinCharsWithQuestionMark

The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `VisualQueryWildcardMinCharsWithAsterisk` property.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. For Quick Search, edit the values of the `WildcardMinCharsWithAsterisk` and `WildcardMinCharsWithQuestionMark` properties.

3. For Visual Query, edit the values of the `VisualQueryWildcardMinCharsWithAsterisk` and `VisualQueryWildcardMinCharsWithQuestionMark` properties.

4. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

5. In a command prompt, navigate to the `toolkit\scripts` directory.

6. Stop Liberty:

```
setup -t stopLiberty
```

7. Update the i2 Analyze application:

```
setup -t deployLiberty
```

8. Start Liberty:

```
setup -t startLiberty
```

Run a selection of Quick Search and Visual Queries that use conditions including wildcard characters. Continue to change the configuration until you are satisfied with the wildcard behavior.

Configuring Visual Query

Analysts can use Visual Query to search for records in the Information Store based on the values of specific properties and on their relationship to other records. You can configure a number of aspects of Visual Query, including the conditions that are available to analysts and how regularly saved queries are run.

You can configure how users complete Visual Queries by modifying the minimum number of characters that must be included in a search term that includes wildcards. For more information, see [“Modifying the wildcard minimum character limits” on page 164](#).

Visual Query condition restrictions

In order to create a configuration file that helps you to manage your Visual Query performance effectively, it is important to understand the available restrictions. Having a proper understanding of the implications of the rules that can be created, allows you to reduce the number of iterations required.

The `visual-query-configuration.xml` file in `configuration/fragments/opal-services/WEB-INF/classes` includes commented out examples of rules applicable to the law enforcement schema. Using these examples as guidance, you can create rules that apply to your system.

Important: The rules that are added in `visual-query-configuration.xml` are applied to the i2 Analyze server and not the Analyst's Notebook Premium client. As such, users are still able to construct queries that include restricted conditions, but the server will prevent the query from being run.

Depending on your circumstances, there are two main approaches that can be used to create your rules:

Invalidating a particular type of query

If you are aware of query types that are an issue, you can deny the specific conditions that lead to the issue, allowing all other Visual Query behavior to remain unchanged.

Enabling specific queries

If you are uncertain about the type of queries that might produce an issue, or want to restrict the types of searches that can be carried out, you can deny all conditions, and selectively allow the types of query that you require.

As you build up your Visual Query configuration file, it is worth bearing in mind the following key concepts:

Rule components

Each rule consists of the following components:

- Rule Type - Whether the rule should allow or deny a specific type of operation
- Item Type - Which Item Type the rule should apply to
- Property Type - Which Property Types for the specified Item Type that the rule should cover
- Operator - Which Operator Types to apply the rule to
- Date and Time Aspects - Which types of temporal aspect to apply the rule to. For example, day of the week, or time of the day.

Implicit 'all valid'

If one or more of the above rule components are not specified, the rule will be applied to all the Visual Query conditions that would be valid.

Rule ordering

Visual Query rules are applied sequentially, meaning that if conflicting rules are added, they are handled in order, allowing rules that are later in the file to overwrite earlier rules. This allows you to refine your conditions, for example:

```
<Deny/>
<!--Allow 'Person' searches to include conditions for
'Date of birth' and 'Gender'
that are exactly or between a specified range -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="EQUAL_TO, BETWEEN"/>
```

Using these rules, searches for people with specified dates of birth or genders are permitted, but all other Visual Query conditions are prevented from running.

Restricting Visual Query conditions

By default, all the Visual Query operators are available for all the record types that support them in your system. To prevent the creation of queries that take too long to produce results, you can configure Visual Query to restrict the operators that are valid in conditions.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Set the rules that you would like to make to your Visual Query conditions:

- a) Using an XML editor, open the `visual-query-configuration.xml` file.
You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.

- b) Using the supplied examples, add the rules that you would like to implement.

Note: Ensure that you add the restrictions paying attention to the ordering as later rules will override earlier rules if applicable.

- c) Save your changes.

2. Update the `DiscoServerSettingsCommon.properties` file to use your configuration rules.

- a) Using a text editor, open the `DiscoServerSettingsCommon.properties` file.

You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.

b) Set the `VisualQueryConfigurationResource`

For example: `VisualQueryConfigurationResource=visual-query-configuration.xml`

c) Save your changes.

Redeploy i2 Analyze to update the application with your changes.

3. In a command prompt, navigate to the `toolkit\scripts` directory.

4. Stop Liberty:

```
setup -t stopLiberty
```

5. Update the i2 Analyze application:

```
setup -t deployLiberty
```

6. Start Liberty:

```
setup -t startLiberty
```

Run a selection of Visual Queries that use conditions. Continue to change the configuration until you are satisfied with the Visual Query condition.

Rule Type

There are two types of Visual Query condition rule. The type of restriction determines whether the operators specified in the rule are allowed or denied.

The type of rule can be either:

- Deny - Specify a rule that prevents an operation
- Allow - Specify a rule that enables an operation

Note: For each rule that you add to the `visual-query-configuration.xml`, you must specify the type of rule.

Item Type Identifier

A restriction that only includes the restriction type will apply to all the types of item that are specified in the schema. If required, to apply the restriction to a specific type of item, you can add an Item Type Identifier.

The `ItemTypeId` attribute allows you to specify an item type to restrict. It assumes that the specified value is found in the current i2 schema that is being used in your deployment. To help ensure that your values are valid, you might want to have the schema open for reference when you are creating your restrictions. In addition, to help troubleshoot issues, it may also help to add the display name or description of the item type into a comment about the restriction.

For example the following item type in the schema:

```
<EntityType Id="ET5"  
SemanticTypeId="guid8A586959-9837-47DE-8DBF-BC7031F01545"  
Description="Person details"  
DisplayName="Person"  
Icon="Person (Shaded Shirt)">
```

could be used to create the following rule in your configuration file:

```
<!--Allow 'Person' searches to include conditions
that are exactly or between a specified range -->
<Allow ItemTypeId="ET5" Operators="EQUAL_TO, BETWEEN"/>
```

Note: You can only specify one type of item per rule.

Property Type Identifiers

If you have added an Item Type Identifier to a Visual Query rule, by default, the rule will apply to all the applicable property types. If required, to apply the rule to specific property types, you can add the property types identifiers.

The PropertyTypes argument allows you to specify the property types to restrict. It assumes that the specified values are found in the current i2 schema that is being used in your deployment. To help ensure that your values are valid, you might want to have the schema open for reference when you are creating your restrictions. In addition, to help troubleshoot issues, it may also help to add the display name or description of the property type into a comment about the rule.

Note: Property Type Identifiers can only be applied to rules that specify an Item Type, and the Property Types must correspond to the specified Item Type

For example the following property types in the law enforcement schema:

```
<EntityType Id="ET5"
SemanticTypeId="guid8A586959-9837-47DE-8DBF-BC7031F01545"
Description="Person details"
DisplayName="Person"
Icon="Person (Shaded Shirt)">
...
  <PropertyType Position="2"
Mandatory="false"
SemanticTypeId="guidFE45F1C4-B198-4111-8123-F42D2CD6419D"
DisplayName="Date of Birth"
Description=""
LogicalType="DATE"
Id="PER9">
  <PossibleValues />
</PropertyType>
  <PropertyType Position="3"
Mandatory="false"
SemanticTypeId="guid7548369B-BA9A-4C4B-AEAD-0CB442EAF27"
DisplayName="Gender"
Description=""
LogicalType="SUGGESTED_FROM"
Id="PER15">
  <PossibleValues>
    <PossibleValue Description="" Value="&lt;Unknown&gt;" />
    <PossibleValue Description="Male" Value="Male" />
    <PossibleValue Description="Female" Value="Female" />
  </PossibleValues>
...
</EntityType>
```

could be used to create the following rule:

```
<!--Allow 'Person' searches to include conditions for  
'Date of birth' and 'Gender'  
that match on an exact value or a range of values -->  
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"  
Operators="EQUAL_TO, BETWEEN"/>
```

Note: This example restriction allows conditions to be run that search for 'equal to' exact values. In addition it includes allowing conditions that search between specified values. As a range of values cannot be determined for a 'suggested from' property type, conditions that are between a specified range will only be enabled for dates of birth.

Date and Time Aspects

When a rule specifies property types with the DATE or DATE_TIME data type, there are a number of options for querying that data. Rules can be applied to specific aspects of temporal information.

DATE_AND_TIME

Apply the rule to conditions that include both a date and time. For example: Searches for people spotted in a specific location at specific time.

DATE

Apply the rule to conditions that focus on a date. For example: Searches for people born on a particular day.

TIME

Apply the rule to conditions that focus on a time. For example: Searches for financial transactions that regularly occur at a set time.

DAY_OF_MONTH

Apply the rule to conditions that focus on the day of the month. For example: Searches for people paid on a specific day of the month.

MONTH

Apply the rule to conditions that focus on the month. For example: Searches for people born within a specific month.

QUARTER

Apply the rule to conditions that focus on the quarter of the year. For example: Searches for financial results.

YEAR

Apply the rule to conditions that focus on a specific year. For example: Searches for people born within a specific year.

DAY_OF_WEEK

Apply the rule to conditions that focus on a specific day of the week. For example: Searches for events that always occur on a Tuesday.

WEEK_OF_YEAR

Apply the rule to conditions that focus on the week of the year as calculated using the ISO week date system. For example weekly sales results.

Operators

You can use operators to specify the types of visual query condition to restrict. If you do not specify an operator, the restriction applies to all valid operations.

STARTS_WITH

Applies an implicit asterisk to the end of the condition. For example, Starts with: Fred is equivalent to 'Fred*', which might match; Fred, Frederick, or Freddie.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for  
'First (Given) Name' that start in a given value -->  
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"  
Operators="STARTS_WITH"/>
```

ENDS_WITH

Applies an implicit asterisk to the start of the condition. For example, Ends with: Fred is equivalent to '*Fred', which might match; Fred, Wilfred, or Alfred.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for  
'First (Given) Name' that end in a given value -->  
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"  
Operators="ENDS_WITH"/>
```

CONTAINS

Applies an implicit asterisk to both the start and the end of the condition. For example, 'Contains: Fred' is equivalent to '*fred*', which could match any of the above terms, but also include Alfredo.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include conditions for  
'First (Given) Name' that contain a given value -->  
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"  
Operators="CONTAINS"/>
```

WILDCARD_PATTERN

An exact match to the specified term that includes wildcard characters. For example, Wildcard pattern: Fr?d is equivalent to 'Fr?d', which matches: Fred, but not Alfredo.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include conditions for  
'First (Given) Name' that match on a wildcard pattern -->  
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"  
Operators="WILDCARD_PATTERN"/>
```

NOT_WILDCARD_PATTERN

Excludes an exact match to the specified term that includes wildcard characters. For example, Not wildcard pattern: Fr?d matches anything aside from: Fr?d.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include conditions for  
'First (Given) Name' that do not match on a wildcard pattern -->  
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"  
Operators="NOT_WILDCARD_PATTERN"/>
```

EQUAL_TO

An exact match to the specified term.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that the operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' and 'Gender'
that match on an exact value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="EQUAL_TO"/>
```

NOT_EQUAL_TO

Exact matches to the specified term should be excluded. For example, Not equal to: Fred matches anything aside from: Fred.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that the operator considers. By default, the limit is 256 characters.

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that do not match on an exact value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="NOT_EQUAL_TO"/>
```

GREATER_THAN

Matches values that are higher than a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE

- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are greater than a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="GREATER_THAN"/>
```

GREATER_THAN_OR_EQUAL_TO

Matches values that are higher than or equal to a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are greater than or equal to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="GREATER_THAN_OR_EQUAL_TO"/>
```

LESS_THAN

Matches values that are less than a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are lower than to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15"
Operators="LESS_THAN"/>
```

LESS_THAN_OR_EQUAL_TO

Matches values that are less than or equal to a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that are lower than or equal to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="LESS_THAN_OR_EQUAL_TO"/>
```

BETWEEN

Matches values that are within a set range.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!--Allow 'Person' searches to include conditions for
'Date of birth' that match on a range of values -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="BETWEEN"/>
```

IS_SET

Matches properties with any populated value.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include searches for 'Date of birth' values
that have been entered -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="IS_SET"/>
```

IS_NOT_SET

Matches properties without a value entered.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!--Allow 'Person' searches to include searches for 'Date of birth' values
that have not been entered -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
Operators="IS_NOT_SET"/>
```

Improving Visual Query performance for multi-line string property types

To improve the performance of Visual Query searches for multi-line string property types when using IBM Db2, you can update the Information Store database. After you update the Information Store, the datatype that is used to store multi-line string property values is changed from a CLOB to a VARCHAR.

Before you begin

- The allowed size of a CLOB column is slightly larger than the size of a VARCHAR column. If a CLOB column contains more than 32,672 bytes (or octets), then it cannot be converted and no other CLOB columns in that table will be converted. Before you attempt to convert the CLOB columns, ensure that all data is less than 32,672 bytes in size.
- You can use a toolkit task to update the multi-line string columns automatically, or you can generate the SQL scripts that update the Information Store database and run those scripts manually.
- If i2 Analyze is deployed with high availability or disaster recovery, the process for updating the Information Store is different. For more information, see [HADR](#).

If you want to update the Information Store using the SQL scripts, see [Updating the Information Store manually](#).

Procedure

1. Open a command prompt on the server, and navigate to the `toolkit\scripts` directory of the i2 Analyze toolkit.
2. Run the following command:

```
setup -t convertClobsToVarchars
```

3. Restart i2 Analyze:

```
setup -t restartLiberty
```

When Liberty starts, the process of updating the multi-line string columns is started. This process works in the same way as the processing that occurs to the Information Store after an upgrade. For more information about this processing, and how to monitor its progress, see: [Information Store processing](#).

While the process is running, data cannot be ingested or uploaded into the Information Store.

Updating the Information Store manually

To generate the database scripts that update the columns so that you can run them manually, run the `convertClobsToVarchars` task with the `--scripts` argument. For example:

```
setup -t convertClobsToVarchars --scripts
```

Note: As part of the script generation process, each table that contains a CLOB column is checked to ensure the data in that CLOB column will fit inside a `VARCHAR(32672)` column. This process alone might take several hours depending on the number of rows in each table and the number of CLOB columns. If you know that there are no CLOB columns that contain data that is too large, you can use the `-p skipSizeCheck` parameter to skip this process.

When you run the command, the SQL scripts are generated in the `toolkit\scripts\database\db2\InfoStore\generated\convert-clobs-to-varchars` directory. There is a script for each item type, for example `0200-0300-clob-to-varchar_ET1.sql`. The item type scripts can be run in parallel.

Use the Db2 command line processor to run the generated SQL scripts.

Data size

When running the `convertClobsToVarchars` toolkit task, the following message is displayed if the data in a CLOB column is more than 32,672 bytes (or octets):

```
The <table name> table contains a CLOB column that is too long to fit into a VARCHAR column.  
Conversion of this table has been abandoned.
```

To remedy the issue, truncate the data in the specified CLOB columns, re-run the `convertClobsToVarchars` toolkit task, and restart the server.

When running the SQL scripts, the following messages are displayed in the toolkit log file if the data in a CLOB column is more than 32,672 bytes (or octets):

```
Column <column name> in table <table name> has a defined length of <n> which is greater than the  
maximum permitted size of <n>. (Skipping table conversion).
```

```
Found: <n> values with length greater than 32,672 in table: <table name>, column: <column name>.
(Skipping table conversion).
```

To remedy the issue, truncate the data in the specified CLOB columns, re-run the `convertClobsToVarchars --scripts` toolkit task, and run the scripts again.

Error handling

The process to convert columns from CLOB to VARCHAR is designed to be re-run in the event of a failure.

When you start the Liberty server, monitor the `console.log` to ensure that the process completes successfully. If an error does occur, the process will continue when you restart the server after you resolve the cause of the error.

If you are running the scripts manually, errors are reported in the Db2 command line processor. You can rerun the same script after you resolve the cause of the error.

The most common error is that the Db2 server runs out of disk space while populating the replica tables that are used to retain data while the data type is updated.

HADR

If the i2 Analyze deployment is in HADR mode, you must update the database by using the following procedure:

1. Backup the primary database
2. Generate the update scripts and run the scripts on the primary database
3. Backup the primary database
4. Disable HADR
5. Restore the primary database onto the secondary server
6. Re-enable HADR

Restricting the length of Visual Query lists

When a Visual Query is constructed by a user, certain conditions allow the values to be specified as a list of values. The length of these lists can be restricted to reduce the size of messages from the client.

By default, the number of items that can be used in a list condition for a visual query condition is set to 10000. If you reduce this value, the number of values that can be applied to conditions of this type is reduced.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

To modify the list length:

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Edit the value of `VisualQueryMaxValuesInList`.
3. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

4. In a command prompt, navigate to the `toolkit\scripts` directory.
5. Stop Liberty:

```
setup -t stopLiberty
```


6. Update the i2 Analyze application:

```
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

Run a selection of Visual Queries that use conditions including lists. Continue to change the configuration until you are satisfied with the lists in your Visual Queries that use conditions.

Visual Query schedule

When the user enables alerting on a saved Visual Query, the query is run automatically on schedule. The default schedule can be changed to suit your performance requirements and the number of Visual Queries that are saved with alerting enabled.

By default, the Visual Queries that are saved with alerting enabled are run once every 24 hours, at 00:00 according to the server time. If the result of a query is different from the last time that the query ran, the user receives an alert with this information.

You can change or override the default schedule, as follows.

- Change the time of day that the saved Visual Queries run. For more information, see [“Changing the daily schedule for Visual Queries”](#) on page 179.
- Create a custom schedule to specify when the saved Visual Queries run. For more information, see [“Creating a custom schedule for Visual Queries”](#) on page 180.

Note: Running Visual Queries on schedule might impact performance for users of the system. The following factors are examples of the characteristics of Visual Queries that can affect performance.

- Number of queries
- Complexity of the queries
- Volume of data searched by the queries

In this situation, consider setting the daily schedule for a quiet period, such as overnight, or setting the custom schedule for a low frequency.

Changing the daily schedule for Visual Queries

You can change the time of day that i2 Analyze runs Visual Queries that are saved with alerting enabled. By default, these queries are run once every 24 hours at 00:00 according to the server time.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Edit the value of `AlertScheduleTimeOfDay`.
The default value is 00:00. The required format is *HH:mm*. *HH* is the hour of the day in a 24-hour time format, 00 - 23, and *mm* is the number of minutes past the hour, 00 - 59.
3. To ensure that the daily schedule value is implemented, check that the `AlertScheduleExpression` property is not enabled.
For more information, see [“Creating a custom schedule for Visual Queries”](#) on page 180.
4. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

5. In a command prompt, navigate to the `toolkit\scripts` directory.

6. Stop Liberty:

```
setup -t stopLiberty
```

7. Update the i2 Analyze application:

```
setup -t deployLiberty
```

8. Start Liberty:

```
setup -t startLiberty
```

After you redeploy i2 Analyze, the Visual Queries that are saved with alerting enabled are run according to the updated daily schedule.

Creating a custom schedule for Visual Queries

To create a custom schedule to run the Visual Queries that are saved with alerting enabled, use a UNIX C cron expression. The default schedule to run these Visual Queries is once every 24 hours, at 00:00 hours according to the server time.

You can create a custom schedule in the `AlertScheduleExpression` property. The `AlertScheduleExpression` property requires a value in UNIX C cron expression format.

With the property `AlertScheduleExpression`, you can override the default daily schedule. By default, this property is not enabled. If the property is enabled, whenever the value of the expression matches the current date and time, i2 Analyze runs the Visual Queries that are saved with alerting enabled. For more information, see [UNIX cron format](#).

When you enable the property `AlertScheduleExpression`, its value overrides the value of the default property, `AlertScheduleTimeOfDay`. For more information about `AlertScheduleTimeOfDay`, see [“Changing the daily schedule for Visual Queries”](#) on page 179.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Uncomment `AlertScheduleExpression`.
When you uncomment the `AlertScheduleExpression` property, the value for this property is used instead of the value for the `AlertScheduleTimeOfDay` property.
3. Edit the value of `AlertScheduleExpression`.
For example, the value `0,4,16,20 * * * *` schedules the Visual Queries that are saved with alerting enabled to run every 4 hours. A short way of expressing this schedule option is `0 */4 * * *`.
4. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

5. In a command prompt, navigate to the `toolkit\scripts` directory.
6. Stop Liberty:

```
setup -t stopLiberty
```

7. Update the i2 Analyze application:

```
setup -t deployLiberty
```

8. Start Liberty:

```
setup -t startLiberty
```

After you deploy i2 Analyze, the updated schedule is applied to run Visual Queries that are saved with alerting enabled.

Highlight queries

In the IBM i2 Enterprise Insight Analysis Investigate Add On, users can search for records in the Information Store, flag records of interest, and find paths in the data between flagged records. However, the web application does not support charting and expanding the records that users find. Instead, when users select a record, they see its information together with lists of other records that are related to it. The appearance and contents of these lists are driven by *highlight queries*.

The role of highlight queries is to navigate the data in the Information Store on the behalf of users, and to present them with answers to questions that they ask most frequently. For example:

- If the subject record is a car, then your users might be interested in its recent owners or its recent sightings.
- If the subject is a cellphone, users might want to know which other phones it calls most often - or *who owns* the phones that it calls most often.
- If the subject is a person, the key information might be their bank accounts; or the bank accounts that the person's accounts transact with; or the owners of those transacting accounts.

Highlight queries run whenever the user opens or refreshes the page that contains the details of a particular record. The first five results from each highlight query appear in *highlight panes* on the same page as the subject record. Each highlight pane also contains a button that the user can click to show more information and more results.

Technical background

Functionally, a highlight query is a predefined, type-specific, multi-level, conditional expand operation. Using the subject entity record as a seed, a highlight query searches the Information Store for entity records of a particular type that are connected to the seed by links of a particular type. After that, it can use those records as the seeds for another, similar search, and so on. The results that users see are the entity records found by the final search that the highlight query performs - which might be the first, or the second, or the sixth.

Because highlight queries are type-specific, they are tightly bound to the Information Store schema. Each entity type in the schema can have its own group of highlight queries that run when the user views a record of that type in the Investigate Add On. Since highlight queries are targeted at finding specific relationships in your data, they are not typically portable between different entity types or different i2 Analyze deployments.

Automatically generated highlight queries

When a deployment of i2 Analyze starts up for the first time, the server automatically generates a set of highlight queries from the Information Store schema. This set contains one highlight query for each entity type in the schema that appears in at least one list of valid link end types. Each highlight query searches for entity-link-entity structures in which the types of the target link and entity are the first valid types for such structures.

For example, imagine a simple schema that defines four entity types (Address, Car, Cellphone, and Person) and four link types (Calls, Owns, Related, and Resides). According to the link type definitions:

- Cars can be linked to addresses, through Resides links
- Cellphones can be linked to other cellphones, through Calls links

- People can be linked to addresses (through Resides links), to cars and cellphones (through Owns links), and to other people (through Related links)

In this scenario, with the automatically generated highlight queries in place, a user who selects a cellphone record sees a highlight pane containing a list of cellphones that the first cellphone has called. (Calls is the only valid link type for cellphones, and Cellphone is the only valid end type.)

If the user selects a person record, they see a highlight pane containing a list of the cars that the person owns. (Owns is the *first* valid link type for people, and Car is the *first* valid end type for such links.)

Custom highlight queries

The automatic mechanism for generating highlight queries is designed only to be a starting point. When you write your own highlight queries, you can make them search for more complex relationships between the subject record and others in the Information Store.

For example, for data that matches the same schema as before, you might enable users to select a person record and see a list of people who reside at the same address (person-address-person), or to whom the subject makes cellphone calls (person-cellphone-cellphone-person).

To create a set of custom highlight queries for your data and your users, you write an XML configuration file that is valid according to an XSD file that is derived from the Information Store schema. Subsequent topics describe how to start, edit, and publish a highlight queries configuration for your deployment of i2 Analyze.

Deploying highlight queries

New, production deployments of i2 Analyze have a set of automatically generated highlight queries that retrieve records with simple relationships to a subject record. To replace this set with custom queries that reflect the structure of your data and the needs of your users, you must write your own highlight queries configuration file.

The highlight queries configuration is one of a group of configuration files that you can modify and send to the i2 Analyze server without the need for system downtime. These files are stored in the `toolkit\configuration\live` directory. The default `highlight-queries-configuration.xml` file instructs the server to generate the automatic highlight query set.

To enable updating the server without the need to restart it, ensure that you have a command-line tool such as **postman** or **curl**.

The procedure for writing a new highlight queries configuration file starts with a download of two files from the server. Through a pair of REST endpoints, you can retrieve an XML file that defines the same set of highlight queries that the server generated automatically, and an XSD file that validates it. With these files as a guide, you can develop your own highlight queries, replace the default file, and upload it to the i2 Analyze server.

Highlight queries are closely aligned with the Information Store schema. You create and edit the highlight queries configuration file in your configuration development environment, alongside [Visual Queries](#) and [search results filters](#).

Note: As described in “[Example highlight queries](#)” on page 193, example deployments of i2 Analyze receive an example set of highlight queries instead of the automatically generated set. To replace the example set with an automatically generated set, replace the highlight queries configuration file in `toolkit\configuration\live` with its equivalent from a `configuration\live` directory under the `toolkit\examples` directory. Then, pick up the procedure below after Step 8.

The first part of the process is to fetch the XSD file that corresponds to the Information Store schema from the server.

1. Open a web browser and connect to the i2 Analyze server at `http://host_name/opa1/doc` to view the REST API documentation.

If you are not logged in to the server, you are prompted to do so. The user must be a member of a group that has the [i2:Administrator](#) permission under command access control.

Note: This requirement is in addition to the other requirements on i2 Analyze users, all of whom must be members of groups that confer an access level for at least one dimension value in each security dimension.

2. Open the section for the GET `v1/admin/highlightqueryconfig/xsd` method, and click **Try it out!** to request the XSD file from the server.

The file is displayed in the **Response Body** field.

3. Save the contents of the field to a file named `highlight-queries-configuration.xsd` in the `toolkit\configuration\live` directory.

Next, you need a highlight queries configuration file to edit. To start from scratch, you can use the `highlight-queries-configuration.xml` file from the `live` directory. Alternatively, you can download the file that defines the automatically generated highlight queries.

4. Open the section for the GET `v1/admin/highlightqueryconfig/automatic` method, and click **Try it out!** to request the XML file from the server.

The file is displayed in the **Response Body** field.

5. Save the contents of the field to a file named `highlight-queries-configuration.xml` in the `toolkit\configuration\live` directory, replacing the existing file.

Edit the configuration file.

6. If you do not already have one, obtain and configure an XSD-aware XML editor, as described in [“Setting up your XSD aware XML editor”](#) on page 304.

7. In the XML editor, open the `toolkit\configuration\live\highlight-queries-configuration.xml` file.

8. Using the [reference](#) and [example](#) information, modify the file to define the highlight queries that your analysts require.

Update the deployment with your changes.

The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

To redeploy your changes by using the deployment toolkit only, see [“Redeploying Liberty”](#) on page 304. You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

9. At the command line, navigate to the `toolkit\scripts` directory.
10. Update the server with your configuration file:

```
setup -t updateLiveConfiguration
```

11. Update the running application by using the reload endpoint:

```
curl -i --cookie-jar cookie.txt -d j_username=user_name  
      -d j_password=password  
      http://host_name/context_root/j_security_check
```

```
curl -i --cookie cookie.txt -X  
      POST http://host_name/context_root/api/v1/admin/config/reload
```



Warning: The reload method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

The server validates your highlight queries as it loads them and returns any errors in its response.

12. If the configuration is invalid, modify the `highlight-queries-configuration.xml` file, and then repeat the process to update the deployment.

Test the new and updated highlight queries.

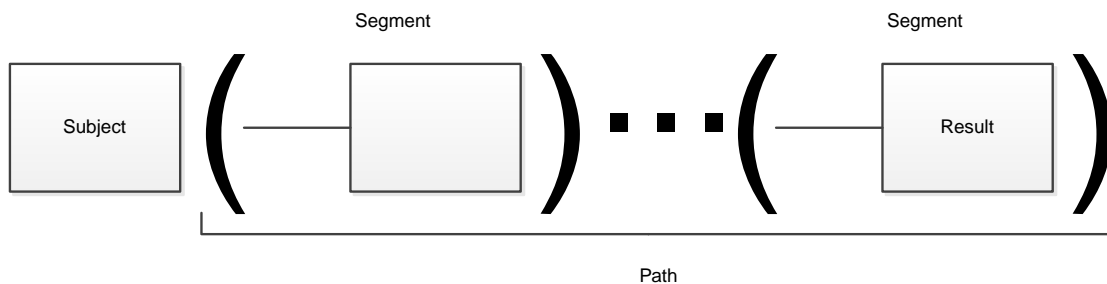
13. In your web browser, view the highlight panes for a record with a number of connections that test your highlight queries.

Structure of highlight queries

The aim of highlight queries is to provide users with relevant information about the records that are associated with the current subject of their investigation. The structure of the queries that you create reflects the idea of starting from the subject and finding records that are one, two, or more links away.

When a user examines records of the same type in the IBM i2 Enterprise Insight Analysis Investigate Add On, they expect to be able to compare them in a consistent way. Highlight queries are grouped by type in the configuration file, and records of a particular type are always displayed with the same set of highlight panes.

Highlight queries themselves use the idea of a *path* to describe how the results of the query are related to the subject. The path is made up of *segments* that take the results one step further from the subject, like this:



At a minimum, each segment in a highlight query specifies the link type and the entity type of the records that match it. For example, a highlight query for finding people who live at the same address as the subject person might have two segments:

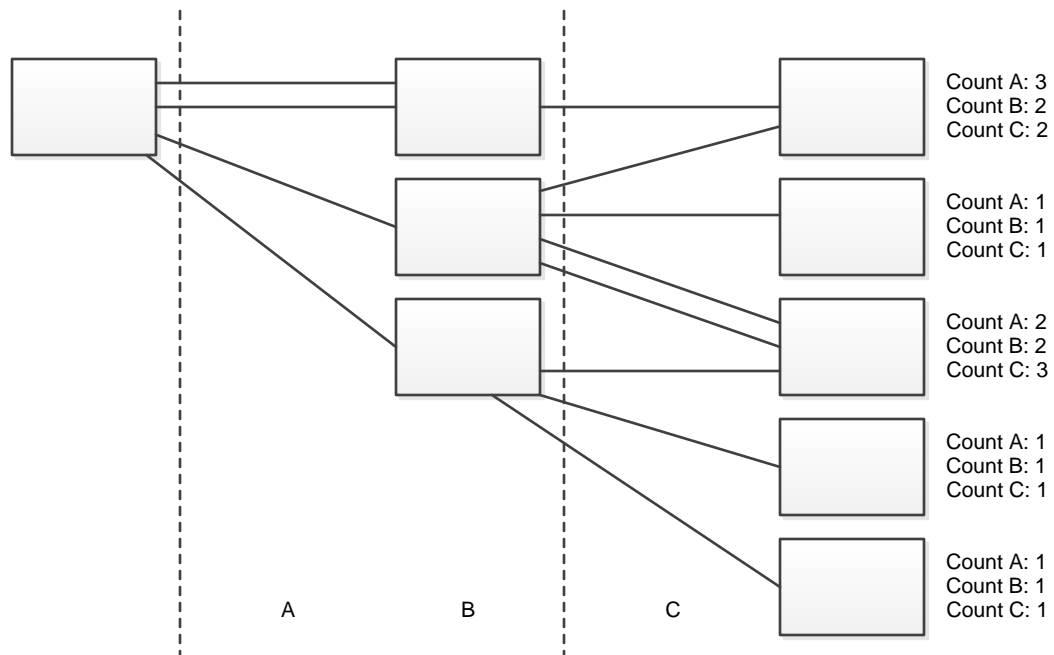
- A link of type 'Lives At' and an entity of type 'Address'. If this segment was the last one in the path, the results would be of type Address, and users might see a list of past and current addresses for the subject.
- A link of type 'Lives At' and an entity of type 'Person'. The records found in the first segment are used as inputs to the second segment, so the full query returns a list of all the people who are known to have lived at the same addresses as the subject.

You can further constrain the results of a highlight query by placing conditions on the link and entity records in the Information Store that each segment matches. Depending on the schema, you might decide that you are only interested in addresses that the subject has 'Lived At' in the last three years, or when the 'Address' is in a particular city or country. Similarly, you might only want to find people who are female (or, alternatively, male).

As well as controlling which records you want to find with a highlight query, you can control how users see the query results in highlight panes. Every query has a name and a description that users can read to understand what the results in front of them mean. For the results themselves, you can specify how to sort them, and what values to display in the limited space that highlight panes provide.

The values that users see in highlight panes do not have to come from the records that form the results of the query. In each segment of a highlight query, you can *export* values from the link and entity records that form part of a found path. For example, imagine a highlight query for bank account records whose single segment finds other accounts that are connected through transaction links. You can export the value of the largest transaction from the segment, and then set it as one of the outputs from the path.

Finally, as well as exporting property values from the segments of a highlight query, you can export (and subsequently output) the counts of the records that the highlight query finds at each point on the path.



In the diagram, the five records on the right are results of a highlight query whose subject was the single record on the left. When you present the results to users, you can output or sort by the counts of records along their paths. The meanings of the counts, and whether they are useful, vary according to the specifics of the highlight query.

The highlight queries configuration file

The permitted XML elements in a highlight queries configuration file include some that are common to all such files, and others that depend upon the Information Store schema. The structure of the file reflects the structure of the highlight queries themselves.

Root element

- `<highlightQueryConfiguration>`

Top-level elements

- [<settings>](#)
- [<highlightQueryGroups>](#)

Highlight query elements

- [<highlightQueryGroup>](#)
- [<highlightQuery>](#)
- [<description>](#)
- [<path>](#)
- [<sortBy>](#)

Path elements

- [<segments>](#)
- [<EntityType>/<LinkType>](#)
- [<direction>](#)
- [<conditions>](#)
- [<exportFields>](#)
- [<outputs>](#)

Condition elements

- [<PropertyType>](#)

Root element

<highlightQueryConfiguration>

`<highlightQueryConfiguration>` is the root element of the highlight query configuration file. In the examples, and in the automatically generated file, it references the `highlight-queries-configuration.xsd` file:

```
<highlightQueryConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="highlight-queries-configuration.xsd">
  ...
</highlightQueryConfiguration>
```

Top-level elements

<settings>

As well as the highlight queries themselves, the configuration file enables control of some aspects of highlight query behavior that can affect the performance of the i2 Analyze server. The optional `<settings>` element supports the following child elements:

- `<maxLinkResultsPerSegment>` - The maximum number of link records that a single segment can match in the Information Store, which defaults to 50000.
- `<maxSeedsPerSegment>` - The maximum number of entity records that any segment except the last can match in the Information Store, which defaults to 1000.
- `<maxResultsPerQuery>` - The maximum number of entity records that the last segment can match in the Information Store, which defaults to 1000.

- `<expireResultsInMinutes>` - The length of time for which the server caches the results of a highlight query, which defaults to 30 minutes.

```
<highlightQueryConfiguration ...>
  <settings>
    <maxLinkResultsPerSegment>50000</maxLinkResultsPerSegment>
    ...
  </settings>
  ...
</highlightQueryConfiguration>
```

If any of the "max" limits is breached, the query in question returns no results and the user sees an error message. Other queries that do not breach the limits continue to run normally.

<highlightQueryGroups>

The only other element permitted as a direct child of `<highlightQueryConfiguration>` is `<highlightQueryGroups>`, which contains all the highlight queries for the deployment, grouped by the entity types of the subject records to which they apply.

```
<highlightQueryConfiguration ...>
  <settings>...</settings>
  <highlightQueryGroups>
    ...
  </highlightQueryGroups>
</highlightQueryConfiguration>
```

Highlight query elements

<highlightQueryGroup>

Each highlight query group contains the highlight queries for subject records of a particular type. The `itemType` attribute of the `<highlightQueryGroup>` element indicates which type the group is for. The valid values for the `itemType` attribute are defined in the XSD file; they are based on the display names of entity types in the Information Store schema.

```
<highlightQueryGroups>
  <highlightQueryGroup itemType="Vehicle-I">
    ...
  </highlightQueryGroup>
  ...
</highlightQueryGroups>
```

<highlightQuery>

A `<highlightQueryGroup>` element can have multiple child `<highlightQuery>` elements. Each `<highlightQuery>` element defines a query to run against the Information Store, and controls some aspects of the highlight pane that displays the query results in the user interface.

The `<highlightQuery>` element supports the `title` attribute whose value is displayed as the title of the highlight pane. It also supports the `automatic` attribute that determines whether the query runs as soon as a user views the subject record. You can specify that users must run a time-consuming query manually (by clicking a button) by setting the value of this optional attribute to `false`.

<description>

In the <description> child of a <highlightQuery> element, you can explain the behavior of the query to users. The contents of this element are displayed to users when they click the information icon (**i**) in a highlight pane heading. At a minimum, IBM recommends that you describe the relationship of the results to the subject record, and the order that you chose to present them in.

<path>

The <path> child of a <highlightQuery> element contains <segments> and <outputs> elements that together provide the logic for the query. Segments are responsible for the [query structure](#), while outputs determine how users see the query results in highlight panes.

<sortBy>

The last permitted child of a [highlightQuery](#) element is <sortBy>. This optional (but important) element specifies how values that were discovered during execution of the query can change the results that users see. For example, depending on your investigation, you might sort a list of people who are associated with a vehicle by their family names or by their dates of association.

The <field> children of <sortBy> are processed in order of appearance. In each one, the id attribute contains the identifier of a field that was exported from the path. The order attribute determines whether results appear in ascending (ASC) or descending (DESC) order of the values of the identified field.

```
<highlightQueryGroup itemType="Vehicle-I">
  <highlightQuery title="People with access" automatic="true">
    <description>People who have access to this vehicle,
      sorted by family name.</description>
    <path>...</path>
    <sortBy>
      <field id="familyName" order="ASC"/>
    </sortBy>
  </highlightQuery>
  ...
</highlightQueryGroup>
```

Path elements

<segments>

<path> elements always have exactly two children: <segments> and <outputs>. <segments> always contains at least one <segment> element that defines a part of the query structure.

```
<path>
  <segments>
    <segment>
      ...
    </segment>
    ...
  </segments>
  <outputs>
    ...
  </outputs>
</path>
```

<segment>

Each <segment> element also has exactly two children that specify the link and entity types that records must have in order to match this part of the query. Like the legal values for the `itemType` attribute of the <highlightQueryGroup> element, the names of these elements are not fixed but depend upon the Information Store schema for the deployment. Furthermore, the link type must be valid for the entity type that you specified in <highlightQueryGroup>, and the entity type must be valid for that link type. For example, a segment for people who have access to vehicles:

```
<highlightQuery title="People with access">
  <path>
    <segments>
      <segment>
        <Access_To-I/>
        <Person-I/>
      </segment>
      ...
    </segments>
  </path>
</highlightQuery>
```

It is valid to use <AnyLinkType/> in place of a specific link type from the schema. It is also valid to use <AnyEntityType/> in place of a specific entity type, in any segment except the last one in the path.

<EntityType>/<LinkType>

The previous example contains a functionally complete segment definition. In a highlight query, the segment finds all the people in the Information Store that are connected to the inputs by 'Access To' links. If the segment is the first in the path, its single input is the subject record. Otherwise, its inputs are the records that the previous segment found.

The "entity type" and "link type" elements that make up a segment support three child elements, all of which are optional. To place further constraints on the records that the segment matches, you can add a <direction> or a <conditions> element. If you want to use some values from the records that you find at this stage of the query to provide context for the final results, you can add an <exportFields> element.

<direction>

For link type elements only, you can constrain matching records according to whether their direction is towards or away from the entity record that forms the input to the segment. The <direction> child of a link type element always contains a single <value> child element whose text content is either INCOMING or OUTGOING.

A constraint of this type is often useful in scenarios such as telephone call analysis or financial investigations, where the direction of a link is key to determining its relevance.

<conditions>

The <conditions> child of an entity type or link type element can contain any number of child elements that correspond to property types of the entity type or link type in question. For each property type, you can place a requirement on the values that matching records must have.

Note: At this version of i2 Analyze, highlight queries do not support conditions on property types with geospatial or multi-line string logical types.

<exportFields>

You might want to export values from a segment of a highlight query for two reasons. First, to display those values to users alongside the results in the highlight pane for the query. Second, to use the values as criteria by which to sort the results of the query.

For a particular highlight query result, the values that you can export are the values of properties from the segments that contributed to that result. For example, imagine a query that finds bank account records that are connected through transaction links. You want to display the transaction dates from those links alongside the found accounts. In this example, every account is connected to the subject record through different links from every other account. If you export the date from the link type, each result sees a date value that comes from its particular links.

```
<segments>
  <segment>
    <Access_To-I>
      <direction>...</direction>
      <conditions>...</conditions>
      <exportFields>
        ...
      </exportFields>
    </Access_To-I>
    <Person-I/>
  </segment>
  ...
```

The <exportFields> element supports any number of two child element types: <propertyField> and <countField>.

<propertyField>

The <propertyField> child of the <exportFields> element has three mandatory attributes:

- **propertyType** - The property type whose value you want to export from the segment. Valid values for this attribute depend on the current entity or link type and are defined in the XSD file. These values are based on the display names of property types in the Information Store schema.
- **aggregatingFunction** - For any particular result, it is possible for a segment to match multiple records. The aggregating function specifies what property value to use when that situation

arises. Set the attribute value to MAX or MIN to select the highest or lowest property value from such a set. Alternatively, set the value to SUM to add the property values from all the matching records together.

- `id` - An identifier for the exported value that you can use to refer to it from elsewhere in the highlight query structure.

```
<Access_To-I>
  <direction>...</direction>
  <conditions>...</conditions>
  <exportFields>
    <propertyField id="typeOfUse" propertyType="type_of_use-P"
      aggregatingFunction="MAX"/>
    ...
  </exportFields>
</Access_To-I>
```

<countField>

As well as using `<propertyField>` to export the highest, lowest, or total value of a property from a set of records, you can use `<countField>` to export the number of records in that set. Returning to the example of linked accounts, it might be useful to tell users how many transactions took place between the subject and each result, or to sort the results by that number.

The `<countField>` element has a single mandatory attribute: `id`, which behaves exactly as it does in `<propertyField>`.

<outputs>

After the `<segments>` in the path are the `<outputs>`, which control the values that users see with the query results in a highlight pane. The `<outputs>` element supports any number of child `<field>` elements, which have two attributes:

- `id` - The identifier of an export field, as previously specified in a `<propertyField>` or `<countField>` element.
- `label` - A value that the user interface uses as a column header for the field.

```
<path>
  <segments>
    <segment>
      ...
    </segment>
    ...
  </segments>
  <outputs>
    <field id="typeOfUse" label="Type of use" />
    ...
  </outputs>
</path>
```

Each highlight pane has space for up to two output fields. If you specify more than two `<field>` elements, users see them only when they click **Show more** in the highlight pane to display more result information.

Condition elements

<PropertyType>

Inside an <EntityType> or <LinkType> element, the <conditions> element can contain any number of child elements whose names are based on property types from the Information Store schema. The child elements restrict the records in the Information Store that match the current segment. The permitted contents of the condition depend on the logical type of the property type.

```
<Access_To-I>
  <conditions>
    <type_of_use-P>
      ...
    </type_of_use-P>
    ...
  </conditions>
  ...
</Access_To-I>
```

Some logical types - geospatial values, multiple-line strings, documents, pictures, and XML data - are not supported in highlight query conditions. According to the generated XSD file, property types with unsupported logical types are not valid children of <conditions> elements. For property types with supported logical types, this version of i2 Analyze supports conditions with four kinds of operators:

- For all valid property types, you can use <isSet> and <isNotSet> elements to specify that matching records must (or explicitly must not) have a value for the property with that type.

```
<conditions>
  <type_of_use-P>
    <isSet/>
  </type_of_use-P>
  ...
</conditions>
```

- For property types whose properties have string values, you can use <equalTo> and <notEqualTo> elements to specify that matching records must have a value for the property that exactly matches (or explicitly does not match) values that you provide.

```
<conditions>
  <type_of_use-P>
    <notEqualTo>
      <value>Passenger</value>
      ...
    </notEqualTo>
  </type_of_use-P>
  ...
</conditions>
```

- For property types whose properties have numeric values, you can use <greaterThan>, <lessThan>, <greaterThanOrEqualTo>, and <lessThanOrEqualTo> elements to specify that

matching records must have a property value with the specified relationship to a value that you provide.

```
<conditions>
  ...
  <transaction_value-P>
    <greaterThan>
      <value>50000</value>
    </greaterThan>
  </transaction_value-P>
  ...
</conditions>
```

- For property types whose properties have date and time values, you can use an `<inThePreceding>` element to specify that matching records must have a value for the property that occurred within a certain period before the current date.

```
<conditions>
  <type_of_use-P>
    ...
  </type_of_use-P>
  <end_date_and_time-P>
    <inThePreceding unit="WEEKS">
      <value>6</value>
    </inThePreceding>
  </end_date_and_time-P>
  ...
</conditions>
```

In `<inThePreceding>` elements, the `<value>` must be an integer, while the `unit` attribute can be one of `WEEKS`, `MONTHS`, or `YEARS`.

If you specify multiple property types in the same `<conditions>` block, their effects are ANDed together. For example, if your conditions are "given name equal to 'John'" and "family name equal to 'Doe'", then only records for which both conditions are true appear in the results.

Conversely, if you use multiple operators against the same property type, their effects are ORed together. For example, you might decide that a property must either be not set or have one of a handful of values. However, you cannot use the same operator more than once, and you cannot use more than one of the numeric operators at the same time. Attempting to do either results in an invalid configuration file.

Example highlight queries

An installation and deployment of i2 Analyze provides two sets of example highlight queries that you can use alongside the documentation when you write your own queries.

The first set of example highlight queries is in the i2 Analyze deployment toolkit.

The `toolkit\examples\highlight-queries\highlight-queries-configuration.xml` file contains an annotated set of highlight queries that are compatible with the example law enforcement schema. In an example deployment of i2 Analyze, these queries take the place of the automatically generated set.

The second set of example highlight queries is in the XML that you can [fetch from a running i2 Analyze deployment](#). These automatically generated queries are compatible with the live Information Store schema.

A worked example

The following listing represents one of the more complicated highlight queries in the toolkit example:

```
<highlightQuery title="Money laundering?" automatic="false">
  <description>Organizations whose accounts transact with an account that also transacts with
    this person's accounts (and might therefore be an intermediary), sorted by value of the
    person's transactions.</description>
  <path>
    <segments>
      <segment>
        <Access_To-I/>
        <Account-I/>
      </segment>
      <segment>
        <Transaction-I>
          <conditions>
            <transaction_currency-P>
              <equalTo>
                <value>US Dollars</value>
              </equalTo>
            </transaction_currency-P>
          </conditions>
          <exportFields>
            <propertyField propertyType="transaction_value-P" aggregatingFunction="MAX"
              id="value"/>
            <propertyField propertyType="date_and_time-P" aggregatingFunction="MAX"
              id="dateTime"/>
            <countField id="transactioncount" />
          </exportFields>
        </Transaction-I>
        <Account-I/>
      </segment>
    </segments>
  </path>
</highlightQuery>
```



```

<segment>
  <Transaction-I>
    <exportFields>
      <countField id="transactioncount2"/>
    </exportFields>
  </Transaction-I>
  <Account-I/>
</segment>
<segment>
  <AnyLinkType/>
  <Organization-I/>
</segment>
</segments>
<outputs>
  <field label="# txns Per -- Inter" id="transactioncount"/>
  <field label="# txns Inter -- Org" id="transactioncount2"/>
  <field label="Largest txn value" id="value"/>
  <field label="Most recent txn date" id="dateTime"/>
</outputs>
</path>
<sortBy>
  <field id="value" order="DESC"/>
</sortBy>
</highlightQuery>

```

This highlight query is for subject records of type 'Person'. The results that it finds are of type 'Organization', because that is the entity type in the final segment. There are four segments in all, which has the potential to make the query resource-intensive. The enclosing `<highlightQuery>` element has its `automatic` attribute set to `false` so that the query only runs when a user requests it.

The first segment in the query finds all the 'Account' records in the Information Store to which the subject is connected through an 'Access To' link. There are no conditions on either part of the segment, and no values are exported for use elsewhere.

The second segment takes all the accounts to which the subject has access, and finds accounts that they have exchanged transactions with. However, it does not find all such accounts, because of the condition on the link type:

```

<conditions>
  <transaction_currency-P>
    <equalTo>
      <value>US Dollars</value>
    </equalTo>
  </transaction_currency-P>
</conditions>

```

The condition restricts the accounts that this segment finds to those where transactions have taken place in US dollars. ("US Dollars" is configured in the Information Store schema as a possible value

for properties of this type.) The next part of the segment then exports information about these dollar transactions for later use:

```
<exportFields>
  <propertyField propertyType="transaction_value-P" aggregatingFunction="MAX"
    id="value"/>
  <propertyField propertyType="date_and_time-P" aggregatingFunction="MAX"
    id="dateTime"/>
  <countField id="transactioncount" />
</exportFields>
```

For each of the eventual results of the highlight query, these lines record the value of the largest transaction at this location in the path, and the date of the most recent transaction. These values might come from different transactions if the count - which we also export here - is greater than one.

The inputs to the third segment, then, are all the accounts that have transacted in dollars with accounts to which the subject has access. The third segment goes on to find any accounts in the Information Store that have exchanged transactions with the inputs. It also exports the count of transactions at this location in the path:

```
<exportFields>
  <countField id="transactioncount2"/>
</exportFields>
```

The final segment takes these accounts that are twice-removed from accounts to which the subject has access, and finds organizations in the Information Store that are connected to them in any way. To do so, it makes use of an `<AnyLinkType>` element:

```
<segment>
  <AnyLinkType/>
  <Organization-I/>
</segment>
```

It is worth recounting what this means from an investigative point of view. For an organization to be found by this query, it must be linked to an account that has transacted with an account that has also exchanged dollar transactions with an account to which the subject has access. In simpler terms, it might be that accounts belonging to the person and the found organizations are exchanging money through third accounts. The query certainly is not conclusive, but the results might become targets for further investigation.

When the results of the query are presented to users, they include the values that were exported from the second and third segments:

```
<outputs>
  <field label="# txns Per -- Inter" id="transactioncount"/>
  <field label="# txns Inter -- Org" id="transactioncount2"/>
  <field label="Largest txn value" id="value"/>
  <field label="Most recent txn date" id="dateTime"/>
</outputs>
```

These lines show the challenges of displaying useful information in the relatively confined space of a highlight pane. In fact, only the first two fields appear in the pane; the others are displayed when the user clicks **Show more** to display more results (and more property values from those results). Ideally,

the labels work in harmony with the <description> of the query that you write to explain the results to your users.

The final part of the highlight query specifies how the application should sort the results. In general, you can request multiple sorts here that are applied in sequence. In this instance, the single criterion is the highest transaction value from the second segment. Setting the order to DESC means that numbers run from high to low, which is the more common requirement. The opposite is true when you sort on text values, and setting order to ASC places the results in alphabetical sequence.

Setting up search results filtering

In the clients of an i2 Analyze server, users can filter search results. You can configure the property types and metadata criteria that appear in the filter list by creating and configuring facets for each type of entity and link record that can appear in search results.

Selecting which property types to use for filtering search results can improve both system performance and the user experience. For example, if the values of properties of a particular type are always unique (license plates, social security numbers), then by default the user sees a separate, one-record facet for every value in the search results. If you configure that property type not to appear, then the system has fewer facets to calculate, and it can display more useful facets from different property types instead.

For records of each entity and link type, the *results configuration file* defines which property types and metadata criteria to use for filtering in Quick Search, external search, and Visual Query results views. All these views display the same set of facets.

The deployment toolkit contains a results configuration file for each of the example schemas in the `examples\schemas` directory, apart from the Chart Store schema. Their names all end in `-results.configuration.xml`. If your system uses a modified version of one of the example schemas, you can modify the appropriate results configuration file.

If you decide to write your own file, the entity and link types that appear must correspond to entity and link types from the deployed schemas. Examine the schemas and the data in your system, and decide which property types to display with facets in the results view. You can also decide which metadata criteria to use for the same purpose.

If you do not specify a results configuration file, all of the property types and metadata criteria that can be displayed with facets, for records of all entity and link types, are displayed in the results view in schema order.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

For more information about the results configuration file and the changes that you can make, see [“Understanding the results configuration file”](#) on page 198.

1. Using an XML editor, open the results configuration file that you want to modify.
2. Add, modify, or remove any <ItemTypeFacet> elements and appropriate child <PropertyTypeFacet> and <MetadataFacet> elements for your deployment.

Note: Property types that have the following logical types cannot be used to filter search results:

- GEOSPATIAL
- MULTIPLE_LINE_STRING

3. Save and close the file.

Note: Ensure that your modified file is stored in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.

4. If you have created the file or changed its name, you must update the name of the file that the deployment uses.
 - a) Using a text editor, open the `DiscoServerSettingsCommon.properties` file in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes` directory.
 - b) Ensure that the value of the `ResultsConfigurationResource` property is set to the name of your results configuration file, and then save and close the file.

Tip: If you do not want to configure search result filtering, clear the value of the `ResultsConfigurationResource` property.

Redeploy i2 Analyze to update the application with your changes.

5. In a command prompt, navigate to the `toolkit\scripts` directory.
6. Stop Liberty:

```
setup -t stopLiberty
```

7. Update the i2 Analyze application:

```
setup -t deployLiberty
```

8. Start Liberty:

```
setup -t startLiberty
```

Run a selection of queries against an i2 Analyze server. Continue to change the configuration until you are satisfied with the types of filter that are available.

Understanding the results configuration file

In the results configuration file, you specify the schema item types, and their associated property types and metadata criteria, by which users can filter search results. By defining the filtering behavior for different entity and link types, you can refine the filtering to make it as valuable as possible for your users.

The results configuration file has the following basic shape:

```
<!-- Results configuration for Law_Enforcement_Schema.xml -->
<tns:ResultsConfiguration ...>
  <Facets InlineLimit="5" ViewAllLimit="100">
    <!-- Address -->
    <ItemTypeFacet TypeId="ET1" Subfacets="ExcludeSpecific">
      <!-- Unique Reference -->
      <PropertyTypeFacet TypeId="ADD1" />
      ...
      <MetadataFacet Criterion="NotesCreatedBy" />
      ...
    </ItemTypeFacet>
    ...
  </Facets>
</tns:ResultsConfiguration>
```

Here, the `<ItemTypeFacet>` element is used to declare which item type you are defining property types and metadata criteria for. The value of the `TypeId` attribute specifies the item type. This value corresponds to the `Id` value of an `<EntityType>` or `<LinkType>` in the schema.

Important: For item types that are defined in gateway or connector schemas, you must augment the item type identifier with the short name of its schema. If the type in the previous example was defined in a gateway schema with the short name External, then the <ItemTypeFacet> element changes:

```
<ItemTypeFacet TypeId="ET1-external" Subfacets="ExcludeSpecific">
  ...
</ItemTypeFacet>
```

In these augmented identifiers, the short name is always in lower-case letters, and any whitespace or non-alphanumeric characters are converted to single hyphens. Furthermore, the short name is always separated from the original identifier with a hyphen.

Property types are specified in child <PropertyTypeFacet> elements. The value of the TypeId attribute specifies the property type. This value corresponds to the Id value of a <PropertyType> in the i2 Analyze schema.

Metadata criteria are specified in child <MetadataFacet> elements. The value of the Criterion attribute specifies the metadata criterion and can have the following values:

FirstUploaded

The date that the record was first uploaded to the Information Store.

FirstUploadedBy

The display name of the user who first uploaded the record.

LastUploaded

The date that the record was last uploaded to the Information Store.

LastUploadedBy

The display name of the user who last uploaded the record.

NotesCreatedBy

The display name of a user who added a note to the record.

SourceRefSourceName

The name of a source that appears in the source references for the record.

SourceRefSourceType

The type of a source that appears in the source references for the record.

IngestionDataSourceName

The data source name that the record was ingested from.

If the record was uploaded from Analyst's Notebook Premium, the IngestionDataSourceName is automatically set to ANALYST.

StorageLocation

For a record that was found through an external search, the location where the record is stored. At this version of the software, the values that users might see are **Information Store** and **Other**.

Note: Any child <MetadataFacet> elements must be specified after all of the child <PropertyTypeFacet> elements within an <ItemTypeFacet> element.

In the <ItemTypeFacet> element, the value of the Subfacets attribute defines the method for specifying the property types and metadata criteria.

The Subfacets attribute can have the following values:

A11

All property types of this item type and metadata criteria are available as filterable options for the results. This behavior is the default if an item type is not added to the results configuration file. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="A11" />
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' and by all the available properties and metadata.

Note: You must not specify any child elements when the value of the Subfacets attribute is A11.

IncludeSpecific

Specific property types and metadata criteria for this item type are displayed in the filtering lists. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="IncludeSpecific">
  <PropertyTypeFacet TypeId="PER4" />
  <PropertyTypeFacet TypeId="PER6" />
  <MetadataFacet Criterion="NotesCreatedBy" />
</ItemTypeFacet>
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' and by 'First (Given) Name', 'Family Name', and 'NotesCreatedBy'.

Note: You must specify at least one child <PropertyTypeFacet> or <MetadataFacet> element when the value of the Subfacets attribute is IncludeSpecific.

ExcludeSpecific

Specific property types and metadata criteria for this item type are excluded from the filtering lists. For example:

```
<ItemTypeFacet TypeId="ET3" Subfacets="ExcludeSpecific">
  <PropertyTypeFacet TypeId="VEH2" />
  <MetadataFacet Criterion="FirstUploaded" />
</ItemTypeFacet>
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Vehicle', but not by 'License Plate Number' or 'FirstUploaded'.

Note: You must specify at least one child <PropertyTypeFacet> or <MetadataFacet> element when the value of the Subfacets attribute is ExcludeSpecific.

None

No property types of this item type and metadata criteria are available for filtering. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="None" />
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' but not by any of the properties of the Person type (such as eye color), nor any of the metadata criteria.

Note: You must not specify any child elements when the value of the Subfacets attribute is None.

Additionally, you can disable all property type and metadata criteria for faceting by using the `PropertyTypeFacetsEnabled` and `MetadataFacetsEnabled` attributes of the `<Facets>` element. By default, both property type and metadata criteria are enabled for faceting. To disable, set the attribute values to `false`.

Allowing users to cancel Expand operations

Some Expand operations against the Information Store can find many i2 Analyze records. Returning all of these records to the user might make the chart too crowded or affect system performance. You can configure i2 Analyze to warn users of large result sets and allow them to cancel the operation.

To specify the number of records that an Expand operation can find before a warning is displayed to analysts, you can provide a value for the `ExpandMaxResultsSoftLimit` setting in the `DiscoServerSettingsCommon.properties` file.

If the number of results exceeds this value, Analyst's Notebook Premium displays a message to inform the user. The user can choose to continue or cancel the operation.

By default this setting has a value of 0, which means that no warning is displayed.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Open the `DiscoServerSettingsCommon.properties` file, and provide a value for the `ExpandMaxResultsSoftLimit` setting.

For example, to display a warning if the number of records exceeds 100:

```
ExpandMaxResultsSoftLimit=100
```

2. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

3. In a command prompt, navigate to the `toolkit\scripts` directory.
4. Stop Liberty:

```
setup -t stopLiberty
```

5. Update the i2 Analyze application:

```
setup -t deployLiberty
```

6. Start Liberty:

```
setup -t startLiberty
```

After you redeploy i2 Analyze, run a selection of Expand queries to test the behavior of the deployment.

Configuring matching

In a deployment of IBM i2 Analyze that contains data that originates from multiple sources, the likelihood of multiple records that represent the same real-world object or relationship increases. To enable analysts to identify and reduce the number of matching records, you can activate the matching features in Analyst's Notebook Premium by creating custom match rules.

i2 Analyze can be presented with data from multiple sources, through multiple routes. The mechanisms that identify matching records depend on how the data is presented to i2 Analyze:

- For records that are on a chart, *system matching* can compare them with records in the Information Store, and *Find Matching Records* can compare records on the chart with each other.
- When data is presented through Analyst's Notebook Premium import, or retrieved by connectors, or created by analysts, system matching can compare incoming records with records on the chart and in the Information Store.
- During the ingestion process, the incoming records are compared with the records in the Information Store. For more information about identifying matches in this way, see [Information Store data correlation](#).

Important:

The scope of match rules is not the same for link records as it is for entity records. Rules for matching entity records apply universally, but rules for matching link records apply only to links between the same pair of ends. Link records that do not connect the same ends are never a match for each other.

This behavior also means that rules for matching link records behave differently in system matching and Find Matching Records:

- In system matching, link records can match only when they are between the same pair of entity records.
- In Find Matching Records, link records can match when they are between any two records in a pair of entity *items* on the chart surface.

If the pair of entity items on the chart contain a large number of united records, some combinations of those records are not searched for matching links. By default, the number of link end record combinations that are searched is 100. You can change this value, but the link matching process might take longer if more combinations are searched. For more information about changing the maximum number of combinations that are searched, see [The DiscoServerSettingsCommon.properties file](#).

Configuring system matching

System matching uses system match rules to identify matching records. To configure system matching, you define and deploy your system match rules on the i2 Analyze server.

The mechanism that you use to deploy system match rules depends on your deployment pattern:

- In a deployment that contains the Information Store, your system match rules are used to create a match index that determines whether a record matches one that exists in the Information Store. The match index is employed by the Get Matches feature in Analyst's Notebook Premium, and when records are presented through import, connectors, or created by analysts. To create the match index from your match rules, see [“Deploying system match rules”](#) on page 203.
- In a deployment that contains only the i2 Connect gateway, a match index is not necessary because the system match rules act only on records on the chart surface. To deploy your system match rules in this deployment pattern, see [“Deploying system match rules for the i2 Connect gateway only”](#) on page 204.

Analysts can choose whether to use system matching at all, but they cannot choose the match rules that are used or approve matches on a record-by-record basis. To ensure that your system match rules do not cause false-positive matches, focus on using known strong identifiers in your data.

If system matching acts on data that involves a large number of seed or matching records (in the tens of thousands), the matching process can take a long time to complete. In this scenario, the following actions might accelerate the process:

- Investigate your match rules to ensure that you are not matching too many records by accident.

- Investigate that the data in your Information Store is being ingested correctly, so that your strong identifiers are not accidentally present on too many records.
- Advise your analysts to use system matching on smaller sets of data.

Deploying system match rules

The system match rules that you deploy are used to create a match index of your data in the Information Store. The match index identifies matches between records that enter the system (or records on a chart) and records in the Information Store.

Ensure that i2 Analyze is started, and that you can connect to the deployment by using Analyst's Notebook Premium.

In your configuration development environment, use Analyst's Notebook Premium to create your match rules, which are saved to an XML file that you can move to the i2 Analyze server. (The match rules files that Analyst's Notebook Premium creates are compatible with both Find Matching Records and system matching, but IBM recommends that you configure the features separately.)

After you create your match rules file, you create a match index of the data in the Information Store from those rules. In fact, every deployment of i2 Analyze has two match indexes, generated from different versions of the match rules file. One of the indexes is live, and the application uses it to process system match requests. The other index is the standby, which is generated when you modify or write new system match rules.

When the standby index is ready, you switch the standby index to live so that the application uses the match index that reflects your new rules.

Note: If your deployment uses only the i2 Connect gateway, follow the instructions in [“Deploying system match rules for the i2 Connect gateway only”](#) on page 204.

1. Connect to your server in Analyst's Notebook Premium to load the i2 Analyze schema, and then create your match rules.

For more information about how to create match rules, see [Find matching records](#) in the Analyst's Notebook documentation.

After you create your rules, Analyst's Notebook Premium saves a file named `local-fmr-match-rules.xml` to the `%localappdata%\IBM i2\Enterprise Insight Analysis\Match Rules\ directory on the local workstation. The file is saved in the directory that was modified most recently.`

Next, you must place the match rules file on the i2 Analyze server and update the deployment with your changes.

2. Move the `local-fmr-match-rules.xml` file to the `toolkit\configuration\environment` directory in the i2 Analyze deployment toolkit.
3. Rename `local-fmr-match-rules.xml` to `system-match-rules.xml`.
4. In a command prompt, navigate to the `toolkit\scripts` directory.
5. Update the system match rules on the server and start the population of the standby match index:

```
setup -t updateMatchRules
```

6. Monitor the `i2analyze\deploy\wlp\usr\servers\opal-server\logs` directory for a `StandbyMatchIndexReady` file. The file is created when the standby match index completes indexing the data in the Information Store with your new system match rules.
7. Update the application to use the standby match index:

```
setup -t switchStandbyMatchIndexToLive
```

This toolkit task switches the standby match index for your new rules to the live index. The match index that was live is now the standby match index.

Connect to the deployment again and test your system rules with representative data to ensure that they meet your requirements. In Analyst's Notebook Premium, you must log out and log in again to see your changes.

If you want to modify the system rules again without using Analyst's Notebook Premium, you can modify the file in an XML editor. For more information about the structure of the match rules file, see [“Match rules syntax” on page 207](#).

Deploying system match rules for the i2 Connect gateway only

In a deployment of i2 Analyze that includes only the i2 Connect gateway, the system match rules identify matches between records from external sources that are added to a chart. There is no match index, and as a consequence the procedure for deploying system match rules is different from systems with an Information Store.

Ensure that i2 Analyze is started, and that you can connect to the deployment by using Analyst's Notebook Premium.

Also, to reload the server through the reload endpoint, ensure that you have a command-line tool such as **postman** or **curl**, and a user that has the `i2:Administrator` command access control permission.

Note: For more information about using the admin endpoints, see [“Using the admin endpoints” on page 305](#).

In your configuration development environment, use Analyst's Notebook Premium to create your match rules, which are saved to an XML file that you can move to the i2 Analyze server. (The match rules files that Analyst's Notebook Premium creates are compatible with both Find Matching Records and system matching, but IBM recommends that you configure the features separately.)

1. Connect to your server in Analyst's Notebook Premium to load the i2 Analyze schema, and then create your match rules.

For more information about how to create match rules, see [Find matching records](#) in the Analyst's Notebook documentation.

After you create your rules, Analyst's Notebook Premium saves a file named `local-fmr-match-rules.xml` to the `%localappdata%\IBM i2\Enterprise Insight Analysis\Match Rules\<data_source_id>` directory on the local workstation. The file is saved in the directory that was modified most recently.

Next, you must place the match rules file on the i2 Analyze server and update the deployment with your changes.

2. Move the `local-fmr-match-rules.xml` file to the `toolkit\configuration\live` directory in the i2 Analyze deployment toolkit.
3. Delete the `system-match-rules.xml` file from the `toolkit\configuration\live` directory.
4. Rename `local-fmr-match-rules.xml` to `system-match-rules.xml`.

Update the deployment with your changes.

The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

To redeploy your changes by using the deployment toolkit only, see [“Redeploying Liberty” on page 304](#). You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

5. In a command prompt, navigate to the `toolkit\scripts` directory.

6. Update the configuration on the server:

```
setup -t updateLiveConfiguration
```

The `updateLiveConfiguration` toolkit task updates server with every file in the `toolkit\configuration\live` directory.

7. Update the running application by using the reload endpoint:

```
curl -i --cookie-jar cookie.txt -d j_username=user_name
      -d j_password=password
      http://host_name/context_root/j_security_check

curl -i --cookie cookie.txt -X POST http://host_name/context_root/api/v1/admin/config/reload
```

To reload the server like this, ensure that you can use the admin endpoints. For more information, see [“Using the admin endpoints”](#) on page 305.



Warning: The reload method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

Connect to the deployment again and test your system rules with representative data to ensure that they meet your requirements. In Analyst's Notebook Premium, you must log out and log in again to see your changes.

If you want to modify the system rules again without using Analyst's Notebook Premium, you can modify the file in an XML editor. For more information about the structure of the match rules file, see [“Match rules syntax”](#) on page 207.

Configuring Find Matching Records

Analysts use Find Matching Records to identify when multiple records on an Analyst's Notebook Premium chart might represent the same real-world object or relationship.

Analysts can write their own match rules for Find Matching Records, but you can also configure i2 Analyze to provide a common set of rules to all analysts when they connect. Analysts can choose whether to use all, none, or a selection of the match rules that the server provides. For more information about how analysts might use Find Matching Records, see [Creating rules for record matching](#).

For information about how to create and deploy server-side match rules for Find Matching Records, see [“Deploying Find Matching Records match rules”](#) on page 205.

Both server and local match rules for Find Matching Records are tied to a particular server. If a destructive schema change in i2 Analyze takes place, all match rules, whether they are deployed on the server or the client, are invalidated.

Deploying Find Matching Records match rules

In a deployment of i2 Analyze, you can deploy match rules on the server for all Analyst's Notebook Premium users to use in Find Matching Records. The match rules are provided to analysts when they connect to the server.

Ensure that i2 Analyze is started, and that you can connect to the deployment by using Analyst's Notebook Premium.

Also, to reload the server through the reload endpoint, ensure that you have a command-line tool such as **postman** or **curl**, and a user that has the `i2:Administrator` command access control permission.

Note: For more information about using the admin endpoints, see [“Using the admin endpoints” on page 305](#).

In your configuration development environment, use Analyst's Notebook Premium to create your match rules, which are saved to an XML file that you can move to the i2 Analyze server. (The match rules files that Analyst's Notebook Premium creates are compatible with both Find Matching Records and system matching, but IBM recommends that you configure the features separately.)

Alternatively, you can create the server rules by editing the match rules XML file manually. If you create the match rules this way, you must still complete the following steps that describe how to configure and deploy the file on the i2 Analyze server.

1. Connect to your server in Analyst's Notebook Premium to load the i2 Analyze schema, and then create your match rules. As you do so, test them with representative data on the chart.

For more information about how to create match rules, see [Find matching records](#) in the Analyst's Notebook documentation.

After you create your rules, Analyst's Notebook Premium saves a file named `local-fmr-match-rules.xml` to the `%localappdata%\IBM i2\Enterprise Insight Analysis\Match Rules\<data_source_id>` directory on the local workstation. The file is saved in the directory that was modified most recently.

Next, you must place the match rules file on the i2 Analyze server and update the deployment with your changes.

2. Move the `local-fmr-match-rules.xml` file to the `toolkit\configuration\live` directory in the i2 Analyze deployment toolkit.
3. Delete the `fmr-match-rules.xml` file from the `toolkit\configuration\live` directory.
4. Rename `local-fmr-match-rules.xml` to `fmr-match-rules.xml`.

Update the deployment with your changes.

The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

To redeploy your changes by using the deployment toolkit only, see [“Redeploying Liberty” on page 304](#). You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

5. In a command prompt, navigate to the `toolkit\scripts` directory.
6. Update the configuration on the server:

```
setup -t updateLiveConfiguration
```

The `updateLiveConfiguration` toolkit task updates server with every file in the `toolkit\configuration\live` directory.

7. Update the running application by using the reload endpoint:

```
curl -i --cookie-jar cookie.txt -d j_username=user_name
      -d j_password=password
      http://host_name/context_root/j_security_check

curl -i --cookie cookie.txt -X POST http://host_name/context_root/api/v1/admin/config/reload
```

To reload the server like this, ensure that you can use the admin endpoints. For more information, see [“Using the admin endpoints” on page 305](#).



Warning: The reload method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

Connect to the deployment again and test that your server rules are visible in the Find Matching Records feature. In Analyst's Notebook Premium, you must log out and log in again to see your changes.

Note: If you copied the `local-fmr-match-rules.xml` file instead of moving it, each rule is duplicated in Analyst's Notebook Premium because the local and server rules are the same.

If you continue to develop your server rules by using this process, the `cached-fmr-match-rules.xml` file in the Match Rules directory contains the rules that are currently deployed on the server. Every time that Analyst's Notebook Premium connects, the cached rules file is overwritten with the latest version from the server.

If you want to modify the server rules again without using Analyst's Notebook Premium, you can modify the file in an XML editor. Any changes that you make are validated when you start i2 Analyze. For more information about the structure of the match rules file, see [“Match rules syntax” on page 207](#).

Match rules syntax

A match rules file is an XML document whose structure is validated when i2 Analyze starts. The match rules syntax is the same for both system match and Find Matching Records match rules files.

Root element: matchRules

The root element of a match rules file is a `<matchRules>` element from the defined namespace. For example:

```
<tns:matchRules
  xmlns:tns="http://www.i2group.com/Schemas/2019-05-14/MatchRules"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation=
    "http://www.i2group.com/Schemas/2019-05-14/MatchRules MatchRules.xsd"
  version="2"
  enableSourceIdentifierMatching="true">
  ...
</tns:matchRules>
```

The `<matchRules>` element has the following customizable attributes:

Attribute	Description
<code>enableSourceIdentifierMatching</code>	For a deployment that contains the Information Store and the i2 Connect gateway, controls whether system matching uses source identifiers to determine whether records match each other, regardless of whether they have property matches. When this attribute is <code>false</code> or absent, different users can cause duplication by retrieving the same records from external sources, editing them, and uploading them to the Information Store. When it is <code>true</code> , such records are detected by system matching.
<code>version</code>	The version of the match rules file, which must be 2 at this release.

matchRule

Inside the root element, each <matchRule> element defines a match rule for records of a particular entity or link type. The <matchRule> element has the following attributes:

Attribute	Description
id	An identifier for the match rule, which must be unique within the match rules file.
itemTypeId	<p>The identifier for the entity or link type to which the rule applies, as defined in the i2 Analyze schema.</p> <p>Important: For item types that are defined in gateway or connector schemas, i2 Analyze appends the schema short name to the item type identifier. For example, if the gateway schema defines an item type with the identifier ET5, then the identifier to use here might be ET5-external.</p> <p>In the modified type identifier, the short name is always in lower-case letters and separated from the original identifier with a hyphen. Any whitespace or non-alphanumeric characters in the short name are converted to single hyphens.</p> <p>When you create or edit match rules through Analyst's Notebook Premium, the application handles these modifications to item type identifiers for you. When you edit the XML file yourself, you are responsible for specifying item type identifiers correctly.</p>
displayName	The name of the rule, which is displayed to analysts in Analyst's Notebook Premium in Find Matching Records.
description	The description of the rule, which is displayed to analysts in Analyst's Notebook Premium in Find Matching Records.
active	Defines whether the rule is active. A value of true means that the rule is active; a value of false means that the rule is not active.
linkDirectionOperator	Determines whether two links must have the same direction in order to match. Mandatory for link type rules, where it must have the value EXACT_MATCH or ANY. Must be absent or null for entity type rules.
version	In earlier versions, the version attribute was mandatory on <matchRule> elements. At this release, the per-rule version is optional, and any value is ignored.

For example, an entity type rule:

```
<matchRule
  id="4a2b9baa-e3c4-4840-a9fd-d204711af50e"
  itemTypeId="ET3"
  displayName="Match vehicles"
  description="Match vehicles with the same license plate when
              either the registered state or region are the same."
  active="true">
  ...
</matchRule>
```

And a link type rule:

```
<matchRule
  id="8aa5f6f4-a1a8-41de-b5c5-8701e44bcde7"
  itemTypeId="LAS1"
  displayName="Match duplicate links"
  description="Match link records between the same pair of entity
              records when the links are in the same direction."
  active="true"
  linkDirectionOperator="EXACT_MATCH">
  ...
</matchRule>
```

matchAll and matchAny

To specify the behavior of a match rule, you can use the following children of the `<matchRule>` element:

matchAll

The `<matchAll>` element specifies that all of the conditions within it must be met.

matchAny

The `<matchAny>` element specifies that at least one of the conditions within it must be met.

A `<matchRule>` element must have both the `<matchAll>` and `<matchAny>` elements. For example:

```
<matchRule ... >
  <matchAll>
    ...
  </matchAll>
  <matchAny />
</matchRule>
```

condition

All match rules must contain the `<matchAll>` and `<matchAny>` elements, although both can be empty for link type rules. It is valid to create a rule that makes all link records of the same type between the same pair of entity records match each other, regardless of any other considerations.

All entity type rules must contain at least one condition. Many link type rules contain conditions too. Each condition defines a comparison that takes place between values in different records, and specifies when those values are considered to match. Conditions can be refined by using operators, values, and normalizations.

To specify the conditions of a match rule, you use the `<condition>` element that can be a child of the `<matchAll>` and `<matchAny>` elements. Each `<condition>` element has a mandatory `propertyTypeId` attribute, which is the identifier for the property type to which the condition applies, as defined in the i2 Analyze schema.

For example:

```
<condition propertyTypeId="VEH2">
  ...
</condition>
```

All conditions contain an <operator> element, most of them a contain <value> element, and many contain <normalizations>.

operator

The <operator> element defines the type of comparison between the property values in different records, or between the property value and a static value specified within the rule. The possible operators are:

Operator	Description
EXACT_MATCH	The values that are compared must match each other exactly.
EXACT_MATCH_START	A specified number of characters at the start of string values must match each other exactly.
EXACT_MATCH_END	A specified number of characters at the end of string values must match each other exactly.
EQUAL_TO	The property values must match each other, and the specified <value>.

For example:

```
<condition propertyTypeId="VEH2">
  <operator>EXACT_MATCH</operator>
  ...
</condition>
```

For more information about the operators that you can use, depending on the logical type of the property, see [Table 1 on page 211](#).

value

The contents of the <value> element affect the behavior of the <operator> of the condition. Different operators require different value types.

- If the operator is EXACT_MATCH_START or EXACT_MATCH_END, the value is an integer that specifies the number of characters to compare at the start or end of the property value:

```
<operator>EXACT_MATCH_START</operator>
<value xsi:type="xsd:int">3</value>
```

- If the operator is EQUAL_TO, the value is a string to compare with the property value:

```
<operator>EQUAL_TO</operator>
<value xsi:type="xsd:string">red</value>
```

- If the operator is EXACT_MATCH, it is not valid to specify a <value> element.

```
<operator>EXACT_MATCH</operator>
```

normalizations

The <normalizations> element contains child <normalization> elements that define how property values are compared with each other (and sometimes with the contents of the <value> element). The possible values for the <normalization> element are:

Normalization	Description
IGNORE_CASE	Ignores case during the comparison ('a' matches 'A')
IGNORE_DIACRITICS	Ignores diacritic marks on characters ('Ã' matches 'A')
IGNORE_WHITESPACE_BETWEEN	Ignores whitespace between characters ('a a' matches 'aa')
IGNORE_WHITESPACE_AROUND	Ignore whitespace around a string (' a ' matches 'a')
IGNORE_NUMERIC	Ignore numeric characters ('a50' matches 'a')
IGNORE_ALPHABETIC	Ignore alphabetic characters ('a50' matches '50')
IGNORE_NONALPHANUMERIC	Ignore non-alphanumeric characters ('a-a' matches 'aa')
SIMPLIFY_LIGATURES	Simplify ligatures ('æ' matches 'ae')

For example, you might have the following normalizations for an EXACT_MATCH operator:

```
<condition ... >
  <operator>EXACT_MATCH</operator>
  <normalizations>
    <normalization>IGNORE_CASE</normalization>
    <normalization>IGNORE_NONALPHANUMERIC</normalization>
    <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
  </normalizations>
</condition>
```

In this example, the values "b m w xdrive" and "BMW x-drive" are considered a match.

The operators and normalizations that you can specify for a condition depend on the logical type of the property type to which the condition applies. The following table shows the operators and normalizations that you can use for each logical type:

Schema logical type	Operators	Normalization
SINGLE_LINE_STRING	All	All
SELECTED_FROM	All	All
SUGGESTED_FROM	All	All
BOOLEAN	EXACT_MATCH	None
INTEGER	EXACT_MATCH	None
DECIMAL	EXACT_MATCH	None
DOUBLE	EXACT_MATCH	None
DATE_AND_TIME	EXACT_MATCH	None
DATE	EXACT_MATCH	None
TIME	EXACT_MATCH	None

Property types that have the following logical types cannot be used in match rules:

- GEOSPATIAL
- MULTIPLE_LINE_STRING

The following XML is an example of a match rules file that contains a single entity match rule. The rule matches vehicle records that have the same values for the license plate property, and the same values for either the state or region properties.

For example, two vehicle records with the license plates "1233 DC 33" and "1233DC33" from the regions "Ile-de-France" and "Île De France" are identified as a match for the following rule:

```

<tns:matchRules
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.i2group.com/Schemas/2019-05-14/MatchRules MatchRules.xsd "
  version="2"
  xmlns:tns="http://www.i2group.com/Schemas/2019-05-14/MatchRules">
<matchRule id="4a2b9baa-e3c4-4840-a9fd-d204711af50e"
  itemTypeId="ET3"
  displayName="Match vehicles"
  description="Match vehicles with the same license plate,
    when either the registered state or
    region are the same."
  active="true">
  <matchAll>
    <condition propertyTypeId="VEH2">
      <operator>EXACT_MATCH</operator>
      <normalizations>
        <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
      </normalizations>
    </condition>
  </matchAll>
  <matchAny>
    <condition propertyTypeId="VEH16">
      <operator>EXACT_MATCH</operator>
      <normalizations>
        <normalization>IGNORE_CASE</normalization>
        <normalization>IGNORE_DIACRITICS</normalization>
        <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
        <normalization>IGNORE_NONALPHANUMERIC</normalization>
      </normalizations>
    </condition>
    <condition propertyTypeId="VEH15">
      <operator>EXACT_MATCH</operator>
      <normalizations>
        <normalization>IGNORE_CASE</normalization>
        <normalization>IGNORE_DIACRITICS</normalization>
        <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
        <normalization>IGNORE_NONALPHANUMERIC</normalization>
      </normalizations>
    </condition>
  </matchAny>
</matchRule>
</tns:matchRules>

```

Configuring geospatial mapping

Analyst's Notebook supports several different mechanisms for mapping chart items, but only one that supports geographic data that is stored in i2 Analyze records. You must define the maps and

geographic reference systems to use before the mapping features that can be used with i2 Analyze records can be enabled.

When records contain geographic information, Analysts can view their location on supplied maps. In addition, Analysts can use visual queries to search for information in specific locations.

The mapping configuration file contains the following main types of information:

Mapping tiles

The mapping configuration file contains information that identifies the map tiles that are available to Analysts. The following details are needed for each map:

- A `displayName` for the mapping tile to display to the Analyst.
- The URL that hosts the mapping tile.
- Any information about the origins of the mapping tile, such as the source and any copyright details.

Base Maps

Base maps form the background on which other GIS items are overlaid. Depending on the type of investigation, and the location, different types of base map might be suitable. For example, you might want to use graphical representations or satellite imagery. In addition to the details that are needed for all mapping tiles, you can specify which base map is opened by default.

Overlays

Layers that sit over the base map and contain detailed information that might be relevant to specific types of investigation. For example, cell tower locations, buildings of interest, borders, or road systems. By adding such information to layers that are overlaid over a base map, you can select the relevant information without displaying information that isn't needed. Unlike base maps, as different overlays can contain different information, multiple overlay tiles can be added to the mapping view. As overlay-mapping tiles are added over other maps, you can set the opacity level to see information on lower mapping tiles.

Coordinate Systems

The 2D coordinate reference systems that can be used to translate coordinates to map locations. The following details are needed for each coordinate system:

- A `displayName` for the coordinate system to display to the Analyst.
- The projection string that defines the coordinate system. This string must be a valid PROJ4 coordinate reference system string.
- The type of editor to use to input the values.

In addition, you can define the bounds to use with the coordinates to constrain the area being mapped.

Defining the mapping solution

In a deployment of i2 Analyze, you can define the mapping tiles that can be accessed, and the coordinate systems that are used to plot locations on them. By default, analysts can select from a list of publicly available maps and common coordinate systems.

Analyst's can access mapping features to help investigate geographic information. You might need to update the mapping tiles that are provided, or the coordinate reference systems that are used to calculate locations. The contents of the example files that are included in the deployment toolkit.

1. Modify the [geospatial-configuration.json](#) file:

- a) In your JSON editor, open the `toolkit\configuration\live\geospatial-configuration.json` file.
- b) Using the [reference](#) and [example](#) information, modify the file to define the mapping tiles and coordinate reference systems your analysts can use.

Update the deployment with your changes.

The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

To redeploy your changes by using the deployment toolkit only, see “Redeploying Liberty” on page 304. You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

2. Update the configuration on the server:

```
setup -t updateLiveConfiguration
```

3. Update the running application by using the admin endpoint:

```
curl -i --cookie-jar cookie.txt -d j_username=user_name -d j_password=password http://  
host_name/conext_root/j_security_check  
  
curl -i --cookie cookie.txt -X POST http://host_name/conext_root/api/v1/admin/config/reload
```

To reload the server by using the reload endpoint, ensure that you can use the admin endpoints. For more information about using the admin endpoints, see [“Using the admin endpoints”](#) on page 305.



Warning: The reload method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

Your configuration is validated, and any errors are reported in the REST response and in the `wlp\usr\servers\opal-server\logs\deployed_war\IBM_i2_Update_Live_Configuration.log`.

Where `deployed_war` can be:

- `opal-services-is`
- `opal-services-daod`
- `opal-services-is-daod`

Depending on the type of spatial data that you are ingesting into your Information Store, to increase the performance of Visual Query operations, you might need to set up spatial database indexes on `IS_DATA` tables that contain geographical data. For more information on creating indexes manually in the Information Store, see [“Creating indexes in the Information Store database”](#) on page 157. In addition, for database specific information about the use of spatial indexes:

- For Db2 databases, see the IBM Integrated Analytics System documentation: [Using indexes and views to access spatial data](#).
- For SQL Server databases, see the Microsoft SQL Server Spatial Indexing documentation: <https://docs.microsoft.com/en-us/sql/relational-databases/spatial/spatial-indexes-overview?view=sql-server-ver15>.

The example mapping configuration file

The toolkit `information-store-opal` and `information-store-daod-opal` example configurations contain an example `geospatial-configuration.json` file in the `configuration/live` folder. This example defines publicly available mapping tiles, and common coordinate reference systems.

The `geospatial-configuration.json` file that is included in the deployment toolkit contains:

Base Map - Esri World Street Map

A link to the publicly available Esri World Street Map tile that presents highway-level data for the world and street-level data for specific regions such as North America, Europe, and Africa. For more information about the Esri World Street Map, see <https://www.arcgis.com/home/item.html?id=3b93337983e9436f8db950e38a8629af>.

Coordinate Systems

The following coordinate systems are included:

OSGB36 National Grid

Used in Great Britain.

NAD83 / UTM zone 10N

Used in North America - between 126°W and 120°W - onshore and offshore. Canada - British Columbia; Northwest Territories; Yukon. United States (USA) - California; Oregon; Washington.

NAD83 / BC Albers

Used in Canada - British Columbia.

WGS 84 / UPS North (N,E)

Used in the northern hemisphere - North of 60°N onshore and off shore, including Arctic.

WGS 84 / UPS South (N,E)

Southern hemisphere - south of 60°S onshore and offshore - Antarctica.

WGS 84 / Pseudo-Mercator

Used in World between 85.06°S and 85.06°N.

You might need to update this file for a number of reasons, for example:

- **Network access** - If you are within a network that is not connected to the internet, you need to host mapping tiles on an internal server, and provide details of your internally hosted mapping tiles. In this scenario, as the example maps are not accessible, the original maps should also be removed.
- **Additional Maps** - You can add references to additional mapping tiles that are either publicly available or hosted locally. This allows you to add different base map tiles to include features such as satellite imagery, and different overlay layers to include items of interest such as key transportation routes or buildings of a particular type.
- **Handling alternative coordinate systems** - If you are aware that coordinates in your system use a particular coordinate reference system, you can update the list to include a different coordinate system that covers a specific area such as the Alberta 3-TM transformation. Alternatively, you can remove coordinate reference systems that are unlikely to be used in your deployment.

The example geospatial-configuration.json file:

```
{
  "mapConfig": {
    "baseMaps": [
      {
        "id": "ESRI_WorldStreetMap",
        "displayName": "ESRI World Street Map",
        "url": "https://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer/
tile/{z}/{y}/{x}",
        "attribution": "Sources: Esri, HERE, Garmin, USGS, Intermap, INCREMENT P, NRCan, Esri
Japan, METI, Esri China (Hong Kong), Esri Korea, Esri (Thailand), NGCC, © OpenStreetMap
contributors, and the GIS User Community"
      }
    ]
  },
  "coordinateSystems": [
    {
      "id": "EPSG:27700",
      "displayName": "OSGB 1936 / British National Grid - United Kingdom Ordnance Survey",
      "projString": "+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000
+ellps=airy +datum=OSGB36 +units=m +no_defs",
      "editorType": "EASTING_NORTHING",
      "bounds": {
        "north": 1300000,
        "east": 700000,
        "south": 0,
        "west": 0
      }
    },
    {
      "id": "EPSG:26910",
      "displayName": "NAD83 / UTM zone 10N",
      "projString": "+proj=utm +zone=10 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs",
      "editorType": "EASTING_NORTHING"
    }
  ],
}
```

```

{
  "id": "EPSG:3005",
  "displayName": "NAD83 / BC Albers",
  "projString": "+proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126 +x_0=1000000 +y_0=0
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs",
  "editorType": "EASTING_NORTHING"
},
{
  "id": "EPSG:32661",
  "displayName": "WGS 84 / UPS North (N,E)",
  "projString": "proj4.defs(\"EPSG:32661\", \"+proj=stere +lat_0=90 +lat_ts=90 +lon_0=0
+k=0.994 +x_0=2000000 +y_0=2000000 +datum=WGS84 +units=m +no_defs\")",
  "editorType": "EASTING_NORTHING"
},
{
  "id": "EPSG:32761",
  "displayName": "WGS 84 / UPS South (N,E)",
  "projString": "+proj=stere +lat_0=-90 +lat_ts=-90 +lon_0=0 +k=0.994 +x_0=2000000
+y_0=2000000 +datum=WGS84 +units=m +no_defs",
  "editorType": "EASTING_NORTHING"
},
{
  "id": "EPSG:3857",
  "displayName": "WGS 84 / Pseudo-Mercator -- Spherical Mercator, Google Maps, OpenStreetMap,
Bing, ArcGIS, ESRI",
  "projString": "+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0
+k=1.0 +units=m +nadgrids=@null +wktext +no_defs",
  "editorType": "EASTING_NORTHING"
}
]
}

```

The geospatial-configuration.json file

The `geospatial-configuration.json` file contains definitions of the mapping tiles and coordinate reference systems to use in your deployment. You can use this reference and example information when you create your own mapping configuration file.

Whether you choose to update the example provided, or create your own file, the `geospatial-configuration.json` contains the following key elements:

- [mapConfig](#)
 - [baseMaps](#)
 - [overlays](#)
- [coordinateSystems](#)

mapConfig

The `mapConfig` object defines the mapping tiles that are available.

The `mapConfig` object can contain a single instance of each of the following objects that are used to define the mapping view and to provide defaults for all mapping layers that are defined in the file:

Option	Description	Type	Default
center	The geographic center of the map when it is first opened. The <code>lat</code> and <code>lng</code> fields are used to specify the center point.	Lat/ Long	Undefined
zoom	The map zoom level when it is first opened.	Number	Undefined
maxBounds	The maximum area that can be mapped by defining the coordinates of two diagonally opposite corners of a rectangle.	Lat/ Long Bounds	Undefined
minZoom	The minimum amount the map can zoom (inclusive). If this option is not specified, the minimum zoom level is calculated as the lowest minimum zoom option available in any defined <code>GridLayer</code> or <code>TileLayer</code> .	Number	
maxZoom	The maximum amount the map can zoom (inclusive). If this option is not specified, the maximum zoom level is calculated as the highest minimum zoom option available in any defined <code>GridLayer</code> or <code>TileLayer</code> .	Number	
wrap	Whether any specified layers are wrapped around the antimeridian. If this option is set to <code>false</code> , the layer is only displayed once at low zoom levels.	Boolean	True

Examples:

```
"mapConfig": {
  "center": {
    "lat": 52.202468,
    "lng": 0.142655
  },
  "zoom": 8,
  "maxBounds": {
    "north": 60.84,
    "south": 49.96,
    "east": 1.78,
    "west": -7.56
  },
  "minZoom": 2,
  "maxZoom": 10,
  "wrap": false,
  ...
}
```

In addition, depending on your mapping requirements, it must contain one or more base maps and can contain one or more overlays:

- [baseMaps](#)
- [overlays](#)

baseMaps

A base map is an image that forms a background over which other GIS items are overlaid. All base maps must be hosted externally, and can be referenced by using a URL. If you would like to reference more than one base map, you can set a default that is used when a map is requested.

Option	Description	Type	Default
id	The unique identifier used to distinguish the layer. (Mandatory)	Text	
displayName	The name to use to refer to this map when the map is displayed. (Mandatory)	Text	
url	<p>The URL template for the tile server that hosts the mapping image. This URL can be a path to a standard image file format: png, or jpg, or a tile layer. (Mandatory)</p> <p>The URL template can contain the following variables:</p> <ul style="list-style-type: none">• {x} and {y} - the tile coordinates• {z} - the zoom level• {s} - substituted with the list of subdomains, a separate request is made for each subdomain• {r} - add "@2x" to load retina tiles <p>Note: i2 Maps use tiles that are based on the EPSG:3857 projected coordinate system (http://epsg.io/3857). Using an unsupported map type does not throw an error, but the map is not rendered correctly.</p>	String	
attribution	<p>Information about the origin of the map that is displayed in the lower right of the map.</p> <p>Hyperlinks are supported alongside other text. You can identify hyperlinks by tagging them in the following format: [hyperlink]url[/hyperlink]</p>	String	
tileSize	The tile size in pixels.	Number	256
minZoom	<p>The minimum amount the map can zoom (inclusive).</p> <p>If this option is not specified, the minimum zoom level is calculated as the lowest minimum zoom option available in the layer.</p>	Number	
maxZoom	<p>The maximum amount the map can zoom (inclusive).</p> <p>If this option is not specified, the minimum zoom level is calculated as the lowest minimum zoom option available in the layer.</p>	Number	

Option	Description	Type	Default
subdomains	Subdomains of the tile service. Can be passed in the form of a string array.	String array	
crossOrigin	Defines how the tile server is configured to handle crossorigin requests. This can be one of the following values: <ul style="list-style-type: none"> anonymous use-credentials "" 	String	Anonymous
tileBounds	The WGS84 bounds for the tiles, allowing you to constrain the tile loading area to a specific location. For example: <pre>tileBounds: { west: 0.72235, east: 0.79101, south: 52.20424, north: 52.35211 }</pre>	Lat/Long Bounds	
defaultBaseMap	If there are multiple base maps, this option identifies the map to load initially.	Boolean	False

Example:

```
"baseMaps": [
  {
    "id": "Esri.DeLorme",
    "displayName": "Esri DeLorme",
    "url": "https://server.arcgisonline.com/ArcGIS/rest/services/Specialty/DeLorme_World_Base_Map/MapServer/tile/{z}/{y}/{x}",
    "attribution": "Tiles © Esri – Copyright: © 2012 DeLorme"
  }
]
```

overlays

An overlay is a type of map layer that is designed to provide additional information in addition to the base map. As these images for a secondary layer over the base map, you can set the opacity of an overlay to allow information present in the base map to be displayed.

Option	Description	Type	Default
id	The unique identifier used to distinguish the layer. (Mandatory)	Text	
displayName	The name to use to refer to this map when the map is displayed. (Mandatory)	Text	

Option	Description	Type	Default
url	<p>The URL template for the tile server that hosts the mapping image. This URL can be a path to a standard image file format: png, or jpg, or a tile layer. (Mandatory)</p> <p>The URL template can contain the following variables:</p> <ul style="list-style-type: none"> • {x} and {y} - the tile coordinates • {z} - the zoom level • {s} - substituted with the list of subdomains, a separate request is made for each subdomain • {r} - add "@2x" to load retina tiles <p>Note: i2 Maps use tiles that are based on the EPSG:3857 projected coordinate system (http://epsg.io/3857). Using an unsupported map type does not throw an error, but the map is not rendered correctly.</p>	String	
attribution	<p>Information about the origin of the map that is displayed in the lower right of the map.</p> <p>Hyperlinks are supported alongside other text. You can identify hyperlinks by tagging them in the following format: [hyperlink]url[/hyperlink]</p>	String	
tilesize	The tile size in pixels.	Number	256
minZoom	<p>The minimum amount the map can zoom (inclusive).</p> <p>If this option is not specified, the minimum zoom level is calculated as the lowest minimum zoom option available in the layer.</p>	Number	
maxZoom	<p>The maximum amount the map can zoom (inclusive).</p> <p>If this option is not specified, the minimum zoom level is calculated as the lowest minimum zoom option available in the layer.</p>	Number	
subdomains	Subdomains of the tile service. Can be passed in the form of a string array.	String array	
crossOrigin	<p>Defines how the tile server is configured to handle crossorigin requests. This can be one of the following values:</p> <ul style="list-style-type: none"> • anonymous • use-credentials • "" 	String	Anonymous

Option	Description	Type	Default
tileBounds	The WGS84 bounds for the tiles, allowing you to constrain the tile loading area to a specific location. For example: <pre>tileBounds: { west: 0.72235, east: 0.79101, south: 52.20424, north: 52.35211 }</pre>	Lat/Long Bounds	
opacity	The degree of visibility for objects on this layer. If set, the opacity must be a decimal between 0.0 (not visible) and 1.0 (fully visible).	Number	1.0

Example:

```
"overlays": [
  {
    "id": "OpenMapSurfer_AdminBounds",
    "displayName": "OpenMapSurfer Admin Bounds",
    "url": "https://maps.heigit.org/openmapsurfer/tiles/adminb/webmercator/{z}/{x}/{y}.png",
    "attribution": "Imagery from [hyperlink]http://giscience.uni-hd.de/ GIScience Research Group @ University of Heidelberg[/hyperlink] | Map data © [hyperlink]https://www.openstreetmap.org/copyright OpenStreetMap[/hyperlink] contributors"
  }
]
```

coordinateSystems

The coordinateSystems object defines the coordinate reference systems that are available.

Option	Description	Type
id	The unique identifier used to distinguish the coordinate system.	Text
displayName	The name to use to refer to this coordinate system when displayed in the UI.	Text
projString	The projection string used to define the coordinate system. This string is a CRS definition.	Text
editorType	The controls to be used to label the x and y axis. This type can be: <ul style="list-style-type: none"> LATITUDE_LONGITUDE - Ellipsoidal 2D CS EASTING_NORTHING - Cartesian 2d CS E,N X_Y - Cartesian 2d CS X,Y 	Text

Option	Description	Type
bounds	The area that can be mapped by defining the coordinates of two diagonally opposite corners of a rectangle. (Optional)	Lat/Long Bounds

Example:

```
"coordinateSystems": [
  {
    "id": "EPSG:3081",
    "displayName": "NAD83 / Texas State Mapping System",
    "projString": "+proj=lcc +lat_1=27.4166666666667 +lat_2=34.9166666666666
+lat_0=31.1666666666667 +lon_0=-100 +x_0=1000000 +y_0=1000000 +ellps=GRS80 +datum=NAD83
+units=m +no_defs",
    "editorType": "X_Y"
  }
]
```

Configuring source references

A source reference provides information about a source that some or all of the data for a record or chart came from. Source references are displayed to analysts in Analyst's Notebook Premium and the Investigate Add On.

You can specify the contents of source references during ingestion, retrieval of data from external sources. Analysts can also create source references in Analyst's Notebook Premium.

A source reference consists of the following fields:

Source name

The name of the source. This is the only field that must contain a value.

For example, "Phone Network".

Source type

The type of the source.

For example, "Cell data".

Source location

The location of the source. This might be a description of a physical location, or a URL to a digital location. If the location value is in a URL format, it is clickable by analysts.

For example, "Filing cabinet 32, drawer 4" or "<http://www.exampleData.com/phone1.html>".

Source image URL

A URL to an image of the source. If the URL resolves to an image, it is displayed with the record in Analyst's Notebook Premium and the Investigate Add On.

For example, "<http://www.exampleImages.com/phone1.jpg>".

Source description

A description of the source.

For example, "This source contains highly accurate information about cell data from the phone network provider".

Specifying the contents of source references

Source references are created or provided in different parts of the system, and you configure the possible values in different ways depending on how the source reference enters the system.

Ingestion source references

The values for source reference name and description are taken from the ingestion source name and description that is specified in the mapping file that is used when you ingest the data.

Records acquire a source reference during the ingestion process. You can provide values for the source reference type, location, and image URL fields for each row in the staging table when you ingest data into the Information Store.

For more information about providing values for this type of source reference, see [Information Store staging tables](#).

Analyst-created source references

Analysts can add source references to records and charts by using Analyst's Notebook Premium. You can constrain the values that analysts can use for the source reference name and type fields.

For more information about configuring these values, see [“Configuring analyst-created source references”](#) on page 225.

Connector source references

When records are retrieved from an external data source via the i2 Connect gateway, the connector implementation can provide a source reference.

For more information about implementing source references into your connectors, see [Returning source references](#).

Configuring analyst-created source references

Analysts can create source references on i2 Analyze records and charts by using the **Record Inspector** or **Chart Inspector** in Analyst's Notebook Premium. By default, analysts can use any values for the source reference name and type, but you can restrict the values that are available for them to use.

It is recommended that you configure analyst-created source references in your configuration development environment before you implement them in your production environment.

You can restrict the values that analysts can specify for the *source name* and *source type* for each item type and for charts. Analysts can provide single-line string values, or you can define selected-from or suggested-from lists that they can choose from. You might choose to restrict the values that analysts can specify to improve the quality of the source references in a system. By enforcing selected-from lists, you can ensure consistent names within your system.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. Enable the use of the source reference schema.
 - a) Create your source-reference-schema.xml file XML schema file in the configuration\fragments\opal-services\WEB-INF\classes directory.
 - b) Specify your source reference schema file name as the value for the SourceReferenceSchemaResource property in the configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties file.
For example:

```
SourceReferenceSchemaResource=source-reference-schema.xml
```

2. Modify the source reference schema configuration file.
 - a) Open your XSD aware XML editor. For more information see [“Setting up your XSD aware XML editor”](#) on page 304.
The associated XSD file is: `toolkit\scripts\xsd\SourceReferenceSchema.xsd`.
 - b) In your XML editor, open your `source-reference-schema.xml` file.
 - c) Using the [reference](#) and [example](#) information, modify the file to define the values that analysts can use.
3. Update the deployment with your changes. On the Liberty server, in a command prompt navigate to the `toolkit\scripts` directory:

```
setup -t deployLiberty  
setup -t startLiberty
```

4. Test the source reference schema that you created.
 - a) In Analyst's Notebook Premium, use the **Record Inspector** or **Chart Inspector** to create source references on records and charts.
In Analyst's Notebook Premium, you must log out and log in again to see your changes.
 - b) Ensure that the values that analysts can use meet your requirements.

Complete this process as many times as necessary until the source reference schema meets your requirements.

The source-reference-schema.xml file

The source reference schema XML file has the following structure. You can use this reference and example information when you create your own source reference schema file.

- [<sourceReferenceSchema>](#)
- [<sourceReferenceSchemaFragments><sourceReferenceSchemaFragment>](#)
- [<sourceName>](#)
- [<sourceType>](#)
- [<possibleValues><possibleValue>](#)

<sourceReferenceSchema>

The `<sourceReferenceSchema>` element is the root of the configuration file.

<sourceReferenceSchemaFragments><sourceReferenceSchemaFragment>

The `<sourceReferenceSchemaFragments>` element can contain one or more `<sourceReferenceSchemaFragment>` child elements.

The `<sourceReferenceSchemaFragment>` defines the values that analysts can use for the source name and source type for a particular set of item types.

The `itemTypeIds` attribute specifies the item types within a set. You must use the item type IDs that come from the i2 Analyze schema for the `itemTypeIds` attribute. You can specify multiple item type IDs as a comma-separated list. If you do not specify any item type IDs, the restrictions apply to all item types that are not referenced elsewhere in the source reference schema.

For example, to make a set for the source references associated with charts and the item type with ID "ET5":


```
<sourceReferenceSchemaFragments>
  <sourceReferenceSchemaFragment itemTypeIds="CHART,ET5">
    ...
  </sourceReferenceSchemaFragments>
```

The `<sourceReferenceSchemaFragment>` can contain one child `<sourceName>` element and one child `<sourceType>` element.

<sourceName>

The `<sourceName>` element contains the possible values that analysts can use as a source name.

The `logicalType` attribute defines whether analysts must choose a value or can provide their own values. The values for the `logicalType` attribute are:

- `SELECTED_FROM` - You provide a list of source names that analysts can use.
- `SUGGESTED_FROM` - You provide a list of source names that analysts can use and they can create their own.
- `SINGLE_LINE_STRING` - Analysts create their own source names.

If you specify the `SELECTED_FROM` or `SUGGESTED_FROM` logical types, use the child `<possibleValues>` element to specify the values that analysts can use.

<sourceType>

The `<sourceType>` element contains the possible values that analysts can use as a source type.

The `logicalType` attribute defines whether analysts must choose a value or can provide their own values. The values for the `logicalType` attribute are:

- `SELECTED_FROM` - You provide a list of source types that analysts can use.
- `SUGGESTED_FROM` - You provide a list of source types that analysts can use and they can create their own.
- `SINGLE_LINE_STRING` - Analysts create their own source types.

The `mandatory` attribute defines whether analysts must specify a value for the source type when they create a source reference. The possible values are `true` and `false`.

If you specify the `SELECTED_FROM` or `SUGGESTED_FROM` logical types, use the child `<possibleValues>` element to specify the values that analysts can use.

<possibleValues><possibleValue>

The `<possibleValues>` element contains child `<possibleValue>` elements. You can specify the `<possibleValues>` element when its parent element specifies `SELECTED_FROM` or `SUGGESTED_FROM` for the `logicalType` attribute.

The `value` attribute of the `<possibleValue>` element contains the value that is displayed to analysts to select.

Example

In this example, analysts can create source references for charts with the source name "Local Police Department" or "Analyst Team 1" to describe the source of the chart.

For item type ET5, for the source name, analysts can use one of the three possible values from the selected-from list. For the source type, they can use one of the three values in the suggested-from list, or provide their own value.

For all other item types, analysts can use any values for the name and type.

```
<tns:sourceReferenceSchema
  xmlns:tns="http://www.i2group.com/Schemas/2019-07-05/SourceReferenceSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.i2group.com/Schemas/2019-07-05/SourceReferenceSchema
SourceReferenceSchema.xsd">
  <sourceReferenceSchemaFragments>

    <sourceReferenceSchemaFragment itemTypeIds="CHART" >
      <sourceName logicalType="SELECTED_FROM">
        <possibleValues>
          <possibleValue value="Local Police Department"/>
          <possibleValue value="Analyst Team 1"/>
        </possibleValues>
      </sourceName>
      <sourceType logicalType="SINGLE_LINE_STRING" mandatory="false">
      </sourceType>
    </sourceReferenceSchemaFragment>

    <sourceReferenceSchemaFragment itemTypeIds="ET5" >
      <sourceName logicalType="SELECTED_FROM">
        <possibleValues>
          <possibleValue value="Police Department"/>
          <possibleValue value="Phone Network"/>
          <possibleValue value="Social Media"/>
        </possibleValues>
      </sourceName>

      <sourceType logicalType="SUGGESTED_FROM" mandatory="true">
        <possibleValues>
          <possibleValue value="Crime Report"/>
          <possibleValue value="RFI - Cell Data"/>
          <possibleValue value="RFI - Social Media"/>
        </possibleValues>
      </sourceType>
    </sourceReferenceSchemaFragment>

    <sourceReferenceSchemaFragment>
      <sourceName logicalType="SINGLE_LINE_STRING">
      </sourceName>
      <sourceType logicalType="SINGLE_LINE_STRING" mandatory="false">
      </sourceType>
    </sourceReferenceSchemaFragment>

  </sourceReferenceSchemaFragments>
</tns:sourceReferenceSchema>
```

Securing i2 Analyze

Depending on the requirements of the deployment, you can configure your deployment to use additional security mechanisms.

Secure Sockets Layer connections with i2 Analyze

Secure Sockets Layer (SSL) technology can be used to establish an encrypted connection between a client and server. You can use SSL to ensure that communication between i2 Analyze components is encrypted.

Depending on your topology and requirements, you can configure SSL for the following connections:

- The client and the HTTP server
- The HTTP server and Liberty
- Liberty, ZooKeeper, and Solr
- Liberty and the database management system

For information about securing the connection between Liberty and any i2 Connectors, see [client authenticated Secure Sockets Layer with the i2 Connect gateway](#).

The version of the TLS protocol that is supported by i2 Analyze is TLS V1.2.

The instructions are intended for readers who are familiar with configuring i2 Analyze, securing network connections, and managing SSL key authentication certificates. Refer also to the documentation for the individual components: IBM HTTP Server, Liberty, Solr, ZooKeeper, Db2, or Microsoft SQL Server.

The instructions are based on a sample scenario for a single-server deployment. The instructions use self-signed certificates to demonstrate working SSL configurations. During the process of creating a production deployment, you can configure SSL in the pre-production environment. In a production deployment, you must use certificates that are signed by a trusted certificate authority. For more information about implementing SSL in a deployment on multiple servers, see [the distributed deployment example on GitHub](#).



Attention: IBM takes reasonable steps to verify the suitability of i2 Analyze for internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. IBM accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

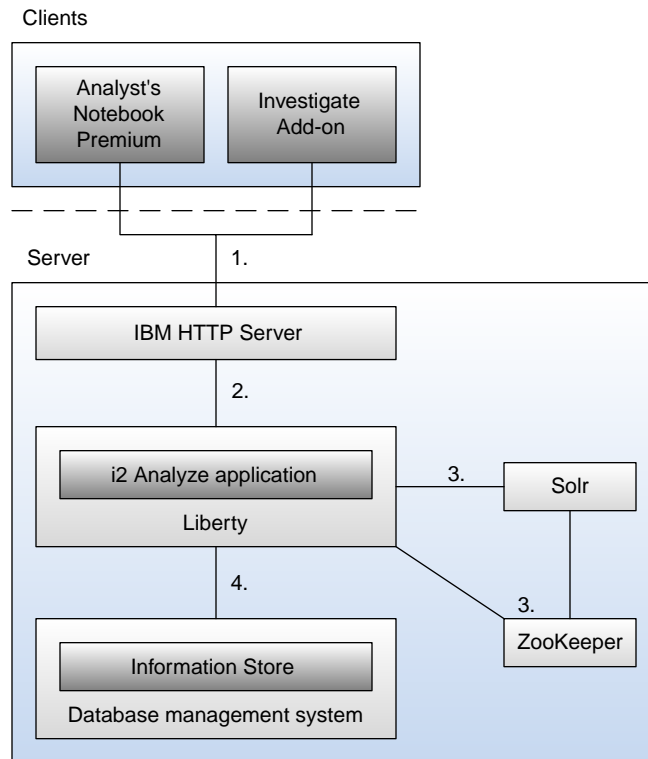
Connections between components of i2 Analyze

Depending on your deployment pattern, i2 Analyze connections can involve the client, the HTTP server, Liberty, the database management system, Solr, and ZooKeeper. You can choose to secure connections by using SSL.

The physical architecture and the network topology of an i2 Analyze deployment determine how appropriate it is to use SSL to secure its connections. For example, if two parts of the deployment are on a single server that you physically control, then the need for SSL to secure the connection between them might be reduced. If your deployment contains a firewall, then you might need to weigh the benefits of inspecting traffic against the benefits of encrypting it.

Note: To enable access to i2 Analyze through the i2 Notebook web client, you *must* use SSL to secure the connection between the browser and the HTTP server. i2 Notebook requires the HTTPS protocol.

The following diagram shows the connections in a deployment that you can use SSL to secure:



The numbered connections in the diagram are as follows:

1. The connections between the clients and the HTTP server.
2. The connection between the HTTP server and Liberty.
3. The connections between Liberty, ZooKeeper, and Solr.
4. The connection between Liberty and the database management system.

To secure connection 2, you must first secure connection 1. To secure connections 3 and 4, you must first secure connections 1 and 2.

SSL certificates for i2 Analyze

SSL communication relies on encryption, keys, and certificates to initiate a secure connection. The certificates are stored in keystore files on the client and the server.

Certificates are exchanged to establish trust during the handshake process that initiates a secure connection. When a certificate is granted through a certificate authority, that certificate can be trusted by the clients or applications that trust certificates that are signed by that authority. A public key certificate that authenticates a server is stored in a keystore file on the server. Trusted certificate authority certificates are stored in the client's truststore file.

As part of the SSL handshake process, certificates are exchanged that are signed by a trusted certificate authority to verify that a certificate is authentic. The environment where you are deploying i2 Analyze might already have a well-defined certificate process that you can use to obtain certificates and populate the required key and truststores.

The examples in the following procedures use self-signed certificates to demonstrate working SSL configurations. In a production deployment, you must use certificates that are signed by a trusted certificate authority.

If all of the public key certificates are signed by the same certificate authority, then you can add the certificate authority's certificate to each of your truststores. If you have a number of certificates to authenticate trust, you might have to add multiple certificates to your truststores.

In the examples, the self-signed certificate is created in a keystore, exported, and imported to the relevant truststore. When you configure SSL communication between components of i2 Analyze, you must have the required certificates in the correct keystores. In the following procedures, examples commands are provided for creating, exporting, and importing self-signed certificates.

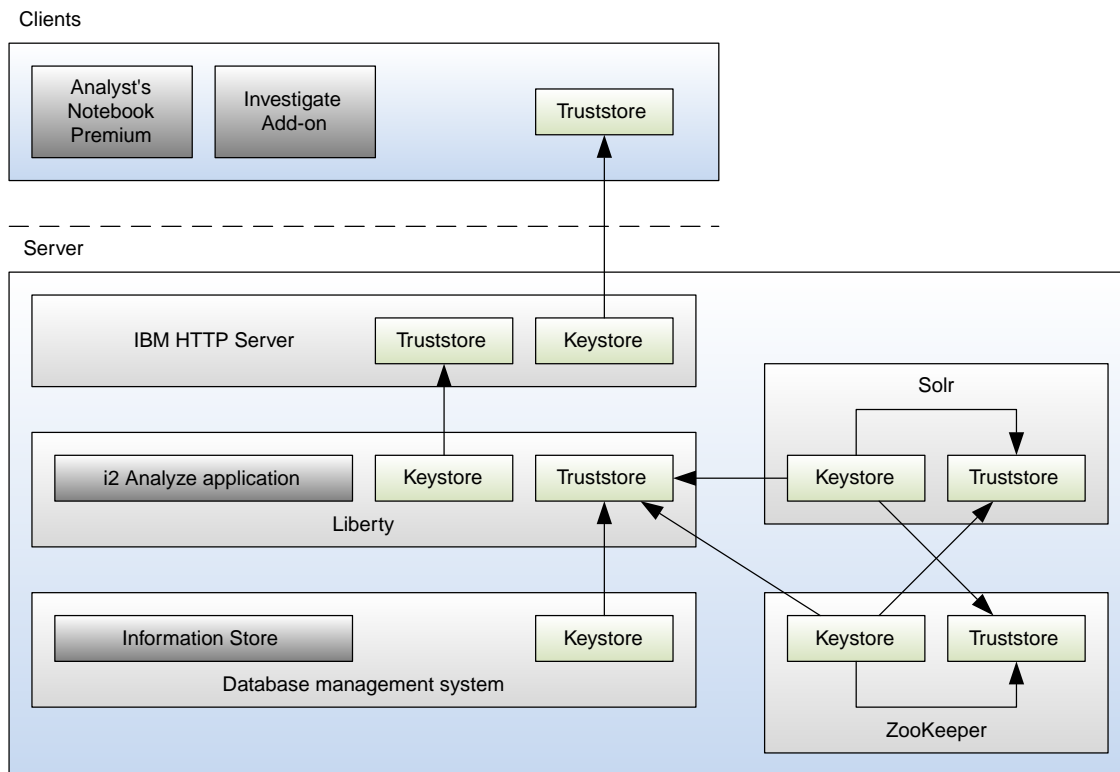
SSL keystores for i2 Analyze

To use SSL to secure a connection between components, i2 Analyze requires Java KeyStore (JKS) or Certificate Management System (CMS) key database files depending on the components. The keystore contains the certificate that acts as the identity of the server and the truststore contains a list of certificates that are trusted by a server.

A JKS file is a repository of security certificates that is used in SSL encryption. In WebSphere Liberty and Solr, a file with extension `.jks` serves as a keystore.

IBM Global Security Kit (GSKit) is a common component that is used by IBM HTTP Server and Db2 for its cryptographic and SSL capabilities. CMS is the native GSKit key database (keystore) that contains certificates. GSKit stores public and private keys and certificates in a key database. A key database consists of a file with a `.kdb` extension and up to three other files with `.sth`, `.rdb`, and `.cr1` extensions. You must save the key database password to a stash file on your computer.

The following diagram shows where these files are used in the components of i2 Analyze and the connections between them.



The environment in which you are deploying i2 Analyze might already contain files that are candidates for use as keystores or truststores. If not, you must create the required files. The files that are required in the sample scenario that is described in the instructions are summarized by component, in the following list.

IBM HTTP Server

To enable SSL connections, the HTTP server requires a CMS key database (*.kdb). The password for this key database must be saved to a stash file. A certificate in the key database identifies the HTTP server and is used when clients connect to i2 Analyze so that they can authenticate the HTTP server. This key database is also used as a truststore to authenticate the certificate that it receives from Liberty.

The i2 Analyze client's truststore, usually located in the operating system or web browser, is used to authenticate the certificate that it receives from the HTTP server.

WebSphere Application Server Liberty

The Liberty server requires a keystore file (*.jks) and a truststore file (*.jks).

A certificate in the keystore identifies the Liberty server and is used to connect to the HTTP server. The HTTP server key database authenticates this certificate that it receives from Liberty.

The certificate in the Liberty truststore is used to authenticate certificates that are received from the database management system keystore and the Solr keystore.

Solr

Each Solr server requires a keystore file (*.jks) and a truststore file (*.jks). Solr requires that all Solr certificates are available in the Solr truststores and the Liberty truststore, so that individual nodes can trust one another, and Liberty can trust the Solr nodes. When you enable ZooKeeper to use SSL, the Solr certificate must also be trusted by the ZooKeeper truststore.

The certificate in the Solr keystore identifies the Solr server and is used to connect to Liberty, and ZooKeeper if it is configured for SSL.

The certificate in the Solr keystore also identifies each Solr node and is used to authenticate secure connection within Solr itself, using the Solr truststore. The certificate in the Solr truststore is used to authenticate the certificate that it received from the Solr keystore.

ZooKeeper

Each ZooKeeper server requires a keystore file (*.jks) and a truststore file (*.jks). ZooKeeper requires that all ZooKeeper certificates are available in the ZooKeeper truststores and the Liberty truststore. This enables the individual servers to trust one another, and Liberty can trust the ZooKeeper servers. Additionally, the Solr certificate must be in the ZooKeeper truststore so that ZooKeeper can trust Solr.

The certificate in the ZooKeeper keystore identifies the ZooKeeper server and is used to connect to Liberty.

The certificate in the ZooKeeper keystore also identifies each ZooKeeper server and is used to authenticate secure connections between the servers in the ZooKeeper quorum, by using the ZooKeeper truststore. The certificate in the ZooKeeper truststore is used to authenticate the certificate that is received from the ZooKeeper keystore.

Database management system

To enable SSL connections, the database management system requires a keystore to connect to Liberty. The type of keystore depends on the type of database management system. For more information about SSL in your database management system, see [“Configuring SSL for a Db2 instance” on page 241](#) or [“Configuring SSL for Microsoft SQL Server” on page 243](#).

The certificate in the database management system keystore identifies the database management system server and is used to connect to Liberty.

In the following procedures, example commands are provided for creating the keystores, certificates, and truststores to use with each component of i2 Analyze. The instructions contain details that are based on a single-server deployment example.

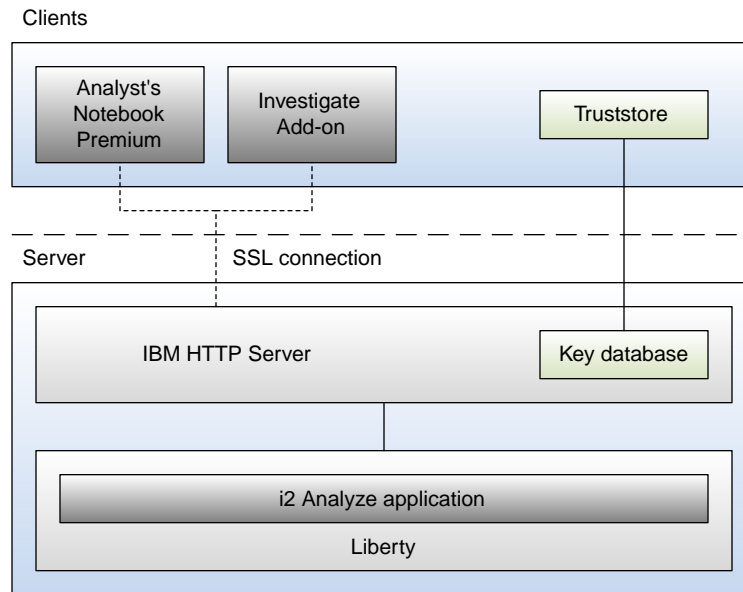
Securing the connection between clients and the HTTP server

i2 Analyze supports the use of SSL to secure the connections between the clients and the HTTP server. After you configure the connections between the clients and the HTTP server, you can opt to configure the connection between the HTTP server and WebSphere Liberty.

Note: To enable access to i2 Analyze through the i2 Notebook web client, you *must* use SSL to secure the connection between the browser and the HTTP server. i2 Notebook requires the HTTPS protocol.

To connect to i2 Analyze by using SSL, the client workstation must first trust the HTTP server because the client connects to i2 Analyze through the HTTP server. Trust is established through a certificate that is associated with the HTTP server and trusted by the client workstation.

The diagram shows the connection that you secure by completing the steps in the following procedures. It also shows the key database that is required.



You must modify the configuration of both the i2 Analyze application and the HTTP server to use SSL to secure the connection.

Creating the HTTP server key database and certificate

The HTTP server stores its associated certificates in a key database. You must create and populate a key database for the HTTP server to use. In a test environment, you can create a self-signed certificate to demonstrate SSL communication.

In i2 Analyze, SSL connections that involve the HTTP server require a key database that contains a signed certificate. In a production deployment, after you create the key database, you must populate it with a certificate that is signed by a trusted certificate authority. To demonstrate a working configuration, you can create and use a self-signed certificate.

The IBM Key Management utility uses a GUI or Window Manager. If you do not have a GUI or Window Manager on your system, you can use the command-line interface.

- For more information about the command-line interface, see [Managing keys from the command line](#).
- For more information about the GUI, see [Managing keys with the IKEYMAN graphical interface](#).

1. Create a key database.

For example, run the following command:

```
gskcapicmd -keydb -create -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -pw "password" -stash
```

- Save a password for the key database to a stash file by using the `-stash` attribute.
- Set `-db` location to the directory that contains the `toolkit` directory in your deployment.

2. Create a self-signed certificate.

For example, run the following command:


```
gskcapicmd -cert -create -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -label "httpKey" -dn "CN=hostname" -san_dnsname "hostname" -pw "password" -sigalg sha256 -size 2048
```

Important: Set the value of CN and san_dnsname to the fully qualified domain name for host name of the server that hosts the HTTP server. The URL that you use to connect to i2 Analyze must use the same value for the host name as the value of the CN. The password is the one that you saved to the stash file in step 1.

3. Extract the certificate from the key database.

For example, run the following command:

```
gskcapicmd -cert -extract -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -label "httpKey" -target "C:\IBM\i2analyze\i2-http-certificate.cer" -pw "password"
```

Set the location of the certificate to the same directory as the key database.

To enable SSL connections to i2 Analyze, the certificate that you added to, or created in, the key database must be installed to be trusted on each client workstation.

Installing the certificate on client workstations

To enable SSL connections to i2 Analyze, the certificate in the key database must be trusted on each client workstation.

The following steps describe how to install and trust the self-signed certificate on a Windows client. To install the certificate on Linux workstations, see the documentation for your operating system.

1. Copy the certificate to any folder on the client workstation.
2. To install the certificate, complete the following steps:
 - a) Double-click the certificate file.
 - b) Click **Install Certificate**, and then click **Next**.
 - c) Click **Place all certificates in the following store**.
 - d) Click **Browse**, select **Trusted Root Certification Authorities**, and click **OK**.
 - e) Click **Next**, and then click **Finish**.

Note: If the certificate is self-signed, Windows displays a security warning because it cannot verify the self-signed certificate. Click **Yes** to accept the certificate.

Configuring the HTTP server for SSL

In a default deployment of i2 Analyze, SSL is unavailable. To enable clients to connect through SSL, you must modify the configuration of both the i2 Analyze application and the HTTP server.

Ensure that the following attributes of the <application> element in the topology.xml are set correctly:

- The http-server-host attribute is set to true
- Ensure that the host-name attribute is set to the name of the server that you are running Liberty on.

For more information about modifying the topology file, see [Modifying the topology](#).

1. Stop the HTTP server.

Run the command to stop the HTTP server, C:\IBM\HTTPserver\bin\httpd -k stop.

2. Edit the http-server.properties file.

- a) Navigate to the toolkit\configuration\environment directory, and open the http-server.properties file in a text editor.

- b) Set the `http.server.ssl.enabled` property to `true`.
 - c) Set the `http.server.keystore.file` property to the location of the key database file of your HTTP server.
For example, `C:/IBM/i2analyze/i2-http-keystore.kdb`.
 - d) Set the `http.server.keystore.certificate.label` property to the label of the HTTP server certificate that is in the key database file. For example, `httpKey`.
3. Open a command prompt and navigate to the `toolkit\scripts` directory.
 4. To deploy i2 Analyze with the edited `http-server.properties` file, and to install the SSL configuration on the HTTP server, run the following commands.

```
setup -t deployLiberty
setup -t configureHttpServer
```

The `httpd.conf` and `plugin-cfg.xml` files are updated to use the SSL configuration.

5. Start Liberty.

To start Liberty, run the following command on each Liberty server:

```
setup -t startLiberty
```

6. Restart the HTTP server.

Run the command to start the HTTP server, `C:\IBM\HTTPserver\bin\httpd -k start`.

If you modified the `FrontEndURI` property in your deployment, you must update it to use the HTTPS protocol. For more information about changing the `FrontEndURI` property, see [“Deploying a proxy server for use with i2 Analyze”](#) on page 73.

To test that the connection is using SSL in your deployment, see [“Testing SSL in an i2 Analyze deployment”](#) on page 236.

Testing SSL in an i2 Analyze deployment

To ensure that a deployment allows only connections that use SSL, you can access it from a client workstation.

- The certificate that the client receives from the HTTP server must be trusted on the client workstation. For more information, see [“Installing the certificate on client workstations”](#) on page 235.
- The Liberty application server must be running.

To connect to i2 Analyze from Analyst's Notebook Premium, you must install the Opal Connector. For more information about installing the Analyst's Notebook Premium, see [Installing IBM i2 Analyst's Notebook Premium](#).

Open Analyst's Notebook Premium and connect to i2 Analyze at the URL `https://host_name/opal`. `host_name` is the fully qualified domain name or IP address of the HTTP server, and matches the **Common Name** value of the certificate.

Note: You cannot connect by using the HTTP protocol `http://host_name/opal`.

When you connect to i2 Analyze, the connection is secure.

Securing connections with Liberty

You can use SSL to secure the connection between WebSphere Liberty and the HTTP server, the database management system, and Solr. To secure connections in i2 Analyze, the HTTP server must

trust the certificate that it receives from Liberty and Liberty must trust the certificates that it receives from the database management system and Solr.

i2 Analyze uses a Java keystore to contain the certificate that is required to identify the i2 Analyze server when it connects with the HTTP server.

i2 Analyze uses a Java truststore to verify the certificates that are received from the database management system, and Solr.

For more information, see [“SSL keystores for i2 Analyze” on page 231](#).

Creating the Liberty keystore and certificate

WebSphere Liberty stores certificates in Java keystore files (.jks). Follow the procedure to create a Java keystore and export the appropriate certificates.

The following steps use a self-signed certificate. In a production environment, use or request a signed certificate for Liberty from a certificate authority. Place this certificate in the Liberty keystore.

Create a keystore and self-signed certificate for Liberty by using the Java keytool utility.

For more information, see [keytool - Key and Certificate Management Tool](#).

- a) Open a command prompt and navigate to the `i2analyze\deploy\java\bin` directory.
- b) Create a keystore and certificate.

For example, run the following command:

```
keytool -genkeypair -alias "libertyKey" -keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks"
-dname "CN=hostname" -keyalg RSA -storepass "password" -ext san=dns:hostname
```

Important: Ensure that you provide values as follows:

- Enter a unique alias.
 - Set the location of the keystore to the directory that contains the toolkit.
 - Set the value of CN and `san=dns` to the hostname of the server that hosts Liberty.
 - Assign a secure password.
- c) Export the certificate from the Liberty keystore.

For example, run the following command:

```
keytool -exportcert -alias "libertyKey" -keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks"
-file "C:\IBM\i2analyze\i2-liberty-certificate.cer" -storepass "password"
```

If you are using self-signed certificates, add the certificate that you exported from your Liberty keystore to the HTTP server key database. For more information, see [“Adding the Liberty certificate to the HTTP key database” on page 239](#).

Configuring Liberty for SSL

To secure the connection between WebSphere Liberty and other components in i2 Analyze, you must configure Liberty for SSL. Update the configuration with the location of a keystore and truststore to use, and the passwords that are used to access the certificates that are contained within them.

Create a keystore and truststore for Liberty that contain the required certificates. For more information, see [“Creating the Liberty keystore and certificate” on page 237](#).

Modify the i2 Analyze `topology.xml` file to specify that a secure connection must be used with the application server. Then, update the `credentials.properties` file to specify the password for the Liberty keystore and truststore files.

When the procedure is complete, it is only possible to connect to Liberty by the HTTPS protocol that uses the secure port that is defined in the port definition properties file. The non-secure port cannot be used.

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

a) In the `<application>` element for the application server to secure, add the `secure-connection` attribute with the value of `true`.

For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
```

Note: The `host-name` attribute value must match the common name that is associated with the certificate for the application server.

b) Add the `<key-stores>` element as a child of the `<application>` element. Then, add a child `<key-store>` element.

For your keystore, specify the type as `key-store` and file as the full path to your keystore.

For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
...
  <key-stores>
    <key-store type="key-store"
      file="C:/IBM/i2analyze/i2-liberty-keystore.jks"/>
  </key-stores>
...
</application>
```

2. Specify the keystore passwords in the credentials file.

a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.

b) Enter the password for the keystore that you specified in the `topology.xml` file.

```
ssl.keystore.password=password
```

3. Update the application with your configuration changes. For more information, see [“Redeploying Liberty”](#) on page 304.

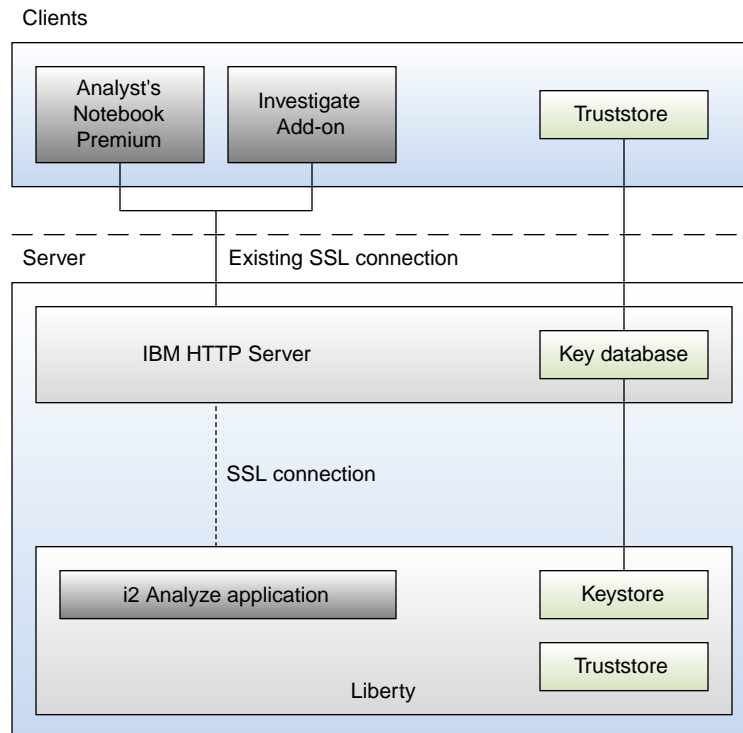
You must secure the connection between the HTTP Server and Liberty. For more information, see [“Securing the connection between the HTTP server and Liberty”](#) on page 238.

Securing the connection between the HTTP server and Liberty

Use SSL to secure the connection between the HTTP server and WebSphere Liberty. The HTTP server key database and the Liberty truststore must contain the required certificates.

- Secure the connection between clients and the HTTP server. For more information, see [“Securing the connection between clients and the HTTP server”](#) on page 233.
- Configure Liberty for SSL. For more information, see [“Configuring Liberty for SSL”](#) on page 237.

The diagram shows the connection that you secure by completing the following instructions. It also shows the key database and keystore that are required.



Adding the Liberty certificate to the HTTP key database

To secure the connection between the HTTP server and Liberty, you must add to the key database the certificate that is required to enable trust with Liberty.

If you are using self-signed certificates, add to the HTTP server key database the certificate that you exported from your Liberty keystore.

In a production environment, import certificates into the HTTP key database to ensure that the certificates that are received from Liberty are trusted.

Add the certificate that you exported from the Liberty keystore into the HTTP server key database.

For more information about exporting the certificate from the Liberty keystore, see [“Creating the Liberty keystore and certificate”](#) on page 237.

For example, run the following command:

```
gskcapicmd -cert -add -db "C:\IBM\i2analyze\i2-http-keystore.kdb" -label "libertyKey" -file
"C:\IBM\i2analyze\i2-liberty-certificate.cer" -pw "password"
```

Configuring IBM HTTP Server

To secure the connection between the i2 Analyze application server and the HTTP server, you must arrange for the `plugin-cfg.xml` file to contain some necessary information.

1. Stop the HTTP server.

2. Navigate to the `toolkit\configuration\environment` directory, and open the `http-server.properties` file in a text editor.
3. Set the value of the `http.server.ssl.require.secure.backend` property to `true`, and then save and close the file.
4. Open a command prompt, and navigate to the `toolkit\scripts` directory.
5. To deploy i2 Analyze with the edited `http-server.properties` file, run the following commands:

```
setup -t configureHttpServer
setup -t deployLiberty
```

The `plugin-cfg.xml` file is updated to enforce that a secure connection is available between the HTTP server and Liberty.

6. Navigate to the `IBM\HTTPServer\plugins\iap\config` directory, and open the `plugin-cfg.xml` file in an XML editor.
7. In each `<ServerCluster>` element, there is a child `<Server>` element. Ensure that each of these `<Server>` elements has a child `<Transport>` element that uses the `https` protocol. Update the `<ServerCluster>` element with the value `"opal-server_cluster"`.
 - a) Add the following element to any of the child `<Server>` elements that do not have a child `<Transport>` element that uses the HTTPS protocol.

```
<Transport Hostname="hostname" Port="portnumber" Protocol="https">
</Transport>
```

hostname is the same as for the `<Transport>` element that uses the HTTP protocol, and *portnumber* matches the value in the port definition properties for the application that you are securing. You can find this value in `C:\IBM\i2analyze\toolkit\configuration\environment\opal-server`.

- b) Add the following `<Property>` elements as children of each `<Transport>` element that uses the HTTPS protocol:

```
<Property Name="Keyring" Value="C:/IBM/i2analyze/i2-http-keystore.kdb"/>
<Property Name="Stashfile" Value="C:/IBM/i2analyze/i2-http-keystore.sth"/>
```

Where the `Value` attributes contain the absolute paths to the keystore for the HTTP server and the associated password stash file.

- c) Save and close the `plugin-cfg.xml` file.

8. Restart the HTTP server.

To ensure that the configuration is correct, look in the `IBM\HTTPServer\plugins\iap\logs\plugin-cfg.log` file.

If the `<Property>` elements for the `keyring` and `stashfile` are not present on each `<Transport>` element in your `plugin-cfg.xml`, the following error message is displayed:

```
ERROR: ws_transport:
transportInitializeSecurity: Keyring was not set.
ERROR: ws_transport:
transportInitializeSecurity: No stashfile or keyring password given.
```

To resolve this issue, ensure that the <Property> elements for the keyring and stashfile are present on each <Transport> element in your plugin-cfg.xml.

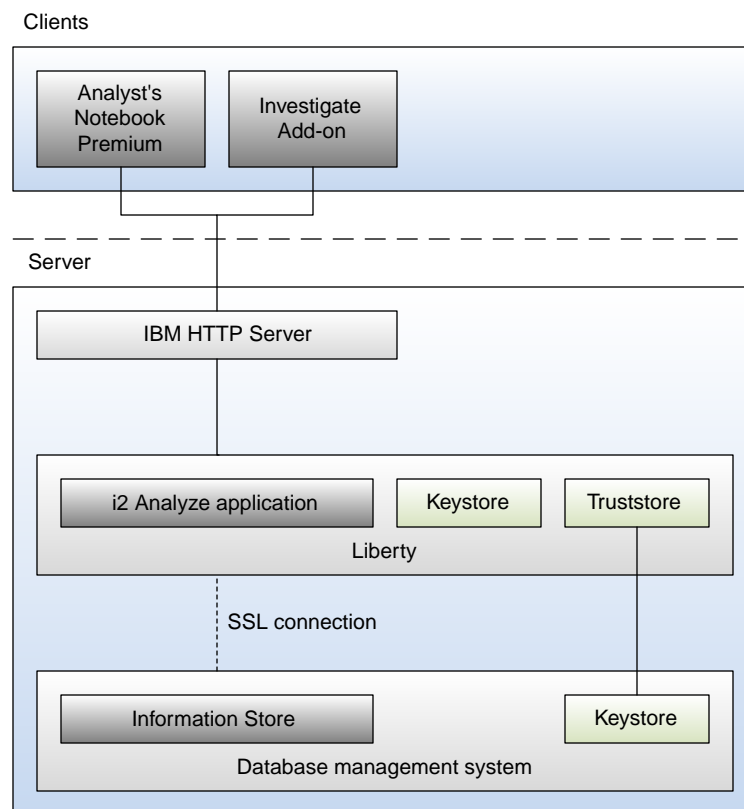
Securing the connection between Liberty and the database

In an i2 Analyze deployment, Liberty connects to the database management system. You can secure the connection between Liberty and the database management system by using SSL.

Configure Liberty for SSL. For more information, see [“Configuring Liberty for SSL”](#) on page 237.

Setting up an SSL connection to the database is a two-part process. First, you must configure your database instance to use SSL. Then, regardless of which database management system you use, you must update the i2 Analyze configuration with the location of the truststore and the truststore password.

This diagram shows the connection that you secure by completing the steps in the following procedures. It also shows the keystore and truststore that are required.



Configuring SSL for a Db2 instance

To secure the connection between the i2 Analyze application server and the database instance, you must change the configuration of both. The Db2 server stores its associated certificates in a key database, so you must create and populate a key database for the Db2 server to use.

In i2 Analyze, SSL connections that involve the Db2 server require a key database that contains a signed certificate. In a production deployment, after you create the key database, you must populate it with a certificate that is signed by a trusted certificate authority. To demonstrate a working

configuration, you can create and use a self-signed certificate. If you are using a certificate that is signed by a certificate authority, then you can add it to the key database and you do not need to complete step 2 to create the self-signed certificate.

Ensure that you understand the details that are provided in the Db2 documentation to configure SSL for your Db2 instance. For more information, see [Configuring Secure Sockets Layer \(SSL\) support in a Db2 instance](#).

To run the commands in the procedure, set your Windows environment variable as follows: set PATH=<path_to_gsk8_directory>\bin;<path_to_gsk8_directory>\lib;%PATH%

1. Create a key database by using the GSKCapiCmd tool.

a) Start the GSKCapiCmd tool.

Note: Start the GSKCapiCmd tool by using the `gskcapicmd` command. Follow the details that are provided in the Db2 documentation link for the path to the command, the required libraries, and the command syntax.

b) Create a key database.

For example, to create the key file, run the following command:

```
gsk8capicmd_64 -keydb -create -db "C:\IBM\i2analyze\i2-db2-keystore.kdb" -pw "password" -stash
```

Important: Ensure that you enter values as follows:

- Set the location of the key database to the directory that contains the toolkit.
- Assign a secure password.

2. Create a self-signed certificate.

For example, to create the self-signed certificate, run the following command:

```
gsk8capicmd_64 -cert -create -db "C:\IBM\i2analyze\i2-db2-keystore.kdb" -label "dbKey" -dn "CN=hostname" -pw "password"
```

Note: The command is a simplified version of the command in the Db2 documentation without the O, OU, L, and C values that are not required for this example. Use a `label` of `dbKey` to align with `httpKey` and `libertyKey` used in the HTTP server and Liberty keystores. Ensure that the common name in the certificate matches the fully qualified domain name of the database instance server.

3. Extract the certificate from the key database.

For example, to extract the certificate, run the following command:

```
gsk8capicmd_64 -cert -extract -db "C:\IBM\i2analyze\i2-db2-keystore.kdb" -label "dbKey" -target "C:\IBM\i2analyze\i2-db2-certificate.cer" -pw "password"
```

4. Configure Db2 for SSL.

For example, to configure Db2 for SSL only, run the following commands:

```
db2 "UPDATE DBM CFG USING SSL_SVR_KEYDB C:\IBM\i2analyze\i2-db2-keystore.kdb"
db2 "UPDATE DBM CFG USING SSL_SVR_STASH C:\IBM\i2analyze\i2-db2-keystore.sth"
db2 "UPDATE DBM CFG USING SSL_SVR_LABEL dbKey"
db2 "UPDATE DBM CFG USING SSL_VERSIONS TLSv12"
db2 "UPDATE DBM CFG USING SSL_SVCENAME 50001"
db2set DB2COMM=SSL
```


5. Navigate to the `toolkit/scripts` directory, and run the command to stop Liberty:

```
setup -t stopLiberty
```

6. Restart Db2 for the changes to take effect:

```
db2stop  
db2start
```

After you configure Db2, you can check the `db2diag.log` file to ensure that there are no errors with your SSL configuration.

Configuring SSL for Microsoft SQL Server

To secure the connection between the i2 Analyze application server and the database instance, you must change the configuration of both. Microsoft SQL Server stores its associated certificates and you must create or obtain certificates for the Microsoft SQL Server to use.

In i2 Analyze, SSL connections that involve SQL Server require i2 Analyze to trust the certificate that it receives from SQL Server. SQL Server stores certificates in the operating system's certificate stores. In a production deployment, you must use a certificate that is signed by a trusted certificate authority. To demonstrate a working configuration, you can create and use a self-signed certificate.

Ensure that you understand the details that are provided in the SQL Server documentation to configure SSL for your SQL Server. For more information, on Windows see [Enable Encrypted Connections to the Database Engine](#) or Linux see [Server Initiated Encryption](#).

Create a self-signed certificate for SQL Server.

1. For example, on Windows you can use the `New-SelfSignedCertificate` command in PowerShell. For information, see [New-SelfSignedCertificate](#).
Run the following command to create a certificate:

```
New-SelfSignedCertificate -Subject "CN=hostname" -KeyAlgorithm RSA  
-KeyLength 2048 -CertStoreLocation "cert:\LocalMachine\My" -TextExtension  
@("2.5.29.37={text}1.3.6.1.5.5.7.3.1") -Hash 'SHA256' -Type SSLServerAuthentication -Provider  
'Microsoft RSA SChannel Cryptographic Provider'
```

Important: Ensure that you set the value of CN to the hostname of the server where SQL Server is located.

2. For example, on Linux you can run the following commands by using OpenSSL:

```
openssl req -x509 -nodes -newkey rsa:2048 -subj '/CN=hostname' -keyout sql-server-key.key -out  
sql-server-certificate.pem -days 365  
sudo chown mssql:mssql sql-server-certificate.pem sql-server-key.key  
sudo chmod 600 sql-server-certificate.pem sql-server-key.key  
sudo mv sql-server-certificate.pem /etc/ssl/certs/  
sudo mv sql-server-key.key /etc/ssl/private/
```

Important: Ensure that you set the value of CN to the hostname of the server where SQL Server is located.

Export the self-signed certificate.

3. On Windows:

- a) Use the **Certificates** snap-in in the Microsoft Management Console to export the certificate from the Local Computer user's certificates.

- b) Locate the self-signed certificate in the Personal certificate store.
- c) **Right-click** the certificate, and click **All Tasks > Export**. Complete the **Certificate Export Wizard** to export the certificate without the private key as a DER encoded binary X.509 file. Set the file name to `i2-sqlserver-certificate.cer`.

For more information about exporting the certificate, see [To export the server certificate](#).

4. On Linux:

- a) Extract the DER certificate from the PEM file by using OpenSSL:

```
openssl x509 -outform der -in sql-server-certificate.pem -out i2-sqlserver-certificate.cer
```

After you create the CER file, ensure that the file permissions are the same as the original PEM file.

Configure SQL Server to encrypt connections.

5. On Windows:

- a) In SQL Server Configuration Manager, expand SQL Server Network Configuration, and right-click **Protocols for <instance>** and click **Properties**.
- b) In the Properties window on the **Certificate** tab, select your certificate from the Certificate list and click **Apply**.
- c) On the **Flags** tab, select **Yes** from the **Force Encryption** list.
- d) Click **OK** and restart the SQL Server instance.

Note: The service account that is used to start the SQL Server instance must have read permissions to your certificate. By default, the service account is `NT Service/MSSQLSERVER` on Windows. For more information about service accounts, see [Service Configuration and Access Control](#).

For more information about encrypted connections, see [To configure the server to force encrypted connections](#).

6. On Linux:

- a) Run the following commands to specify your certificate and key, and configure SQL Server:

```
systemctl stop mssql-server
cat /var/opt/mssql/mssql.conf
sudo /opt/mssql/bin/mssql-conf set network.tls-cert /etc/ssl/certs/sql-server-certificate.pem
sudo /opt/mssql/bin/mssql-conf set network.tls-key /etc/ssl/private/sql-server-key.key
sudo /opt/mssql/bin/mssql-conf set network.tls-protocols 1.2
sudo /opt/mssql/bin/mssql-conf set network.force-encryption 1
```

- b) Restart the SQL Server instance.

Configuring i2 Analyze to connect to a database instance using SSL

To connect to a database instance by using SSL, i2 Analyze must be able to trust and verify the certificate that it receives from the database server. Those requirements mean that the certificate must also be stored in a truststore that i2 Analyze can access.

- Ensure that you configured Liberty for SSL. For more information, see [“Configuring Liberty for SSL” on page 237](#).
- Your database management system must be configured for SSL. For more information, see:
 - [“Configuring SSL for a Db2 instance” on page 241](#)
 - [“Configuring SSL for Microsoft SQL Server” on page 243](#)

i2 Analyze uses a Java truststore to verify the certificate from the database server, and so you must create a truststore on your i2 Analyze server that contains the trusted certificates for your database. You can use the same truststore that you created for Liberty. For more information, see [“Creating the Liberty keystore and certificate” on page 237](#).

For Liberty to communicate with the secured database, in the topology database element you must specify the `secure-connection` attribute to be true and the name of the truststore that contains the database certificate. Also, specify the correct port number, which corresponds to the SSL port for the database. In the `credentials.properties` file, the correct password for the specified truststore must be added.

1. Create the Liberty truststore and import into the truststore the certificate that you exported from the database management system.

For example, run one of the following commands:

If you are using Db2:

```
keytool -importcert -alias "dbKey" -file C:\IBM\i2analyze\i2-db2-certificate.cer -keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks" -storepass "password"
```

If you are using SQL Server:

```
keytool -importcert -alias "dbKey" -file C:\IBM\i2analyze\i2-sqlserver-certificate.cer -keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks" -storepass "password"
```

Enter yes in response to the query, Trust this certificate?

Important: Ensure that you enter values as follows:

- Enter a unique alias.
 - Set the location of the keystore to the directory that contains the toolkit.
 - Assign a secure password.
2. You must register the `i2-database_management_system-certificate.cer` certificate that you exported from the database management system on your i2 Analyze server.
 - On Windows, import the certificate into the Trusted Root Certification Authorities store for the current user.
 - On Linux, copy the certificate to the `/etc/pki/ca-trust/source/anchors` directory and use **update-ca-trust** to enable it as a system CA certificate.
 3. Modify the i2 Analyze topology to use SSL for its database connection.
 - a) In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
 - b) In the `<database>` element for the database that you want to connect to with SSL, add the `secure-connection="true"` attribute.
 - c) In the same `<database>` element, add the `trust-store` attribute with the location of the truststore.

The sample scenario uses a common truststore for Liberty that you created in step 1 and define in the `<key-stores>` element as shown in the example. Alternatively, you can use a specific truststore that is defined for the database management system.

If you are using Db2:

```
<database database-type="InfoStore" dialect="db2" instance-name="DB2"
  database-name="ISTORE" xa="false" id="infostore" host-name="hostname" port-number="50001"
  secure-connection="true" trust-store="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
```

If you are using SQL Server:

```
<database database-type="InfoStore" dialect="sqlserver"
  database-name="ISTORE" xa="false" id="infostore" host-name="hostname" port-number="1433"
  secure-connection="true" trust-store="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
```

d) In same element, ensure that the following attribute values are correct:

- The `host-name` attribute value must match the common name that is associated with the certificate for the database.
- The `port` attribute value must match the value of the port number when you configured the database management system for SSL.

4. Specify the truststore password in the credentials file:

- a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
- b) Enter the password for the truststore that contains the certificate in the `db.infostore.truststore.password` credential.

Update the application with your configuration changes. For more information, see [“Redeploying Liberty” on page 304](#).

Testing the deployment

To test the SSL connection between i2 Analyze and the database management system, connect to i2 Analyze. After you connect, ensure that you can create, browse, and search for data.

1. Connect to your data store by using one of the supported clients. For more information, see [Connecting clients](#).
2. Create, browse, and search for data to ensure that the database connection is working.

Securing the connection between Liberty, Solr, and ZooKeeper

You can secure the connection between Liberty, Solr, and ZooKeeper by using SSL.

Configure Liberty for SSL. For more information, see [“Configuring Liberty for SSL” on page 237](#).

Create the Liberty truststore. For more information, see [“Configuring i2 Analyze to connect to a database instance using SSL” on page 244](#).

Setting up an SSL connection to Solr and ZooKeeper is a three-part process. First, configure Solr to use SSL, then configure ZooKeeper to use SSL, and finally update the i2 Analyze configuration with the location of the Solr and ZooKeeper keystores and passwords.

The certificate in the Solr keystore is used to identify the Solr server. The certificate in the Liberty truststore is used to authenticate certificates that are received from the Solr keystore.

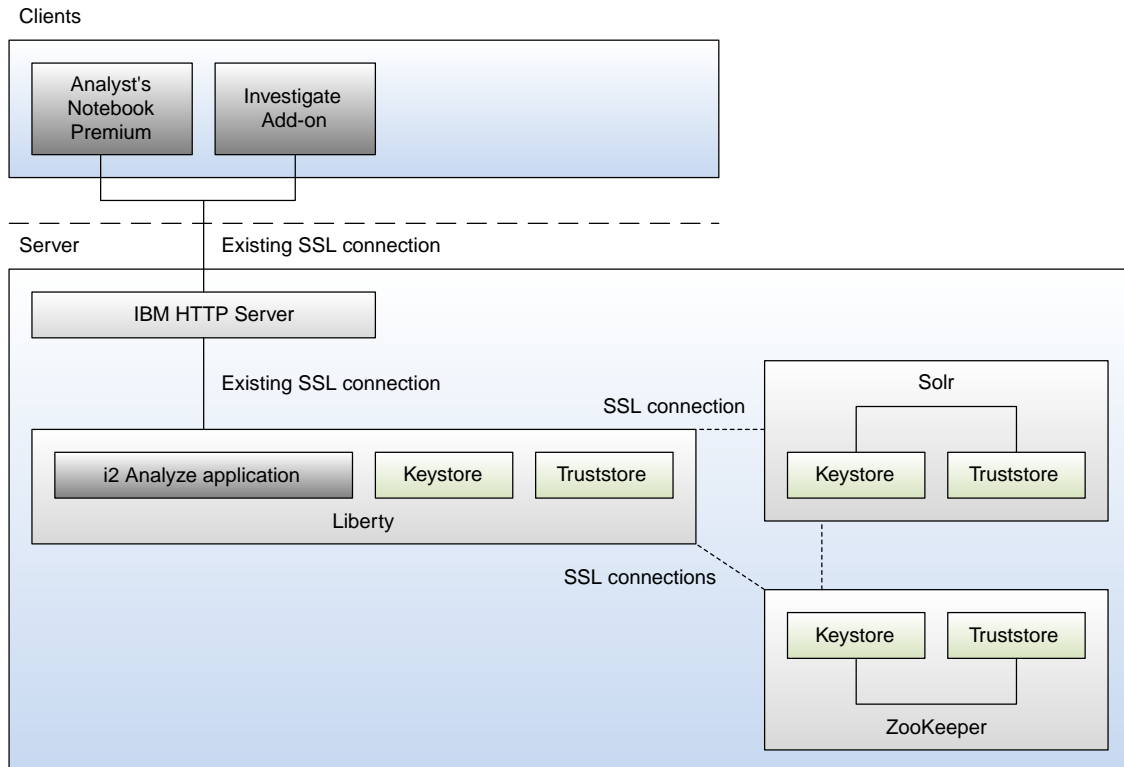
The certificate in the Solr keystore is also used for authentication within Solr itself, using the Solr truststore. The certificate in the Solr truststore is used to authenticate the certificate that it received from the Solr keystore.

The certificate in the ZooKeeper keystore is used to identify the ZooKeeper server. The certificate in the Liberty truststore is used to authenticate certificates that are received from the ZooKeeper keystore.

The certificate in the ZooKeeper keystore is also used for authentication within a ZooKeeper quorum, by using the ZooKeeper truststore. The certificate in the ZooKeeper truststore is used to authenticate the certificate that it received from the ZooKeeper keystore.

The certificate in the ZooKeeper truststore is used to authenticate certificates that are received from the Solr keystore.

The diagram shows the connection that you can secure by completing the following instructions. It also includes the keystores and truststores that are required for a single server.



Creating Solr keystores and certificates

Solr stores its associated certificates in keystore and truststore files (.jks). The certificate in the Solr keystore identifies the server that Solr is deployed on. This certificate is checked against the certificate in the Liberty truststore when Liberty attempts to connect to Solr.

To ensure that communication between the i2 Analyze application server and the Solr index is secured, create a keystore and truststore for Solr.

The following steps use a self-signed certificate. In a production environment, use or request a signed certificate for Liberty from a certificate authority. Place this certificate in the Liberty keystore.

For Solr to work correctly, there must be a keystore for each server on which Solr is deployed. Each keystore contains a certificate that identifies the server. Solr also requires that all Solr certificates are available in the Solr truststore and the Liberty truststore, so that individual nodes can trust one another, and Liberty can trust the Solr nodes.

Depending on the topology of your deployment, you might need to create a separate keystore and truststore on each Solr server.

1. Create a keystore and export a certificate for Solr by using the Java keytool utility.

For more information, see [keytool - Key and Certificate Management Tool](#).

- a) Create a keystore and a certificate.

For example, run the following command:

```
keytool -genkeypair -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-solr-keystore.jks"
-dname "CN=hostname" -keyalg RSA -storepass "password"
```

Important: Ensure that you enter values as follows:

- Set the value of CN to the host name of the server that hosts Solr.
- Set the location of the key database to the directory that contains the toolkit.
- Set the storepass attribute to be a secure password for the keystore.

Enter the key password for the Solr key or press the Return key to confirm that the key password is the same as keystore password.

- b) Export the certificate from the Solr keystore.

For example, run the following command.

```
keytool -exportcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-solr-keystore.jks"
-file "C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
```

Enter the password that you set for the keystore in the previous step.

2. Create the Solr truststore and import the required certificates.

If you are using a self-signed certificate, import the certificate that you exported from the Solr keystore in step 1b.

For example, run the following command:

```
keytool -importcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-solr-truststore.jks" -file
"C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
```

In response to the prompt, enter yes to trust the certificate.

Creating ZooKeeper keystores and certificates

ZooKeeper stores its associated certificates in keystore and truststore files (.jks). The certificate in the ZooKeeper keystore identifies the server that ZooKeeper is deployed on. This certificate is checked against the certificate in the Liberty truststore when Liberty connects to ZooKeeper.

To ensure that communication between Liberty and ZooKeeper is secured, create a keystore and truststore for ZooKeeper.

The following steps use a self-signed certificate. In a production environment, use or request a signed certificate for each component from a certificate authority.

For ZooKeeper to work correctly, there must be a keystore and truststore for each server on which ZooKeeper is deployed. Each keystore contains a certificate that identifies the server. ZooKeeper also requires that each ZooKeeper server's certificate is trusted by each ZooKeeper truststore and the Liberty truststore, so that individual servers can trust one another, and Liberty can trust the ZooKeeper servers.

Depending on the topology of your deployment, you might need to create a separate keystore and truststore on each ZooKeeper server.

1. Create a keystore and export a certificate for ZooKeeper by using the Java keytool utility.

You must complete this for each server on which ZooKeeper is deployed.

For more information, see [keytool - Key and Certificate Management Tool](#).

a) Create a keystore and a certificate.

For example, run the following command:

```
keytool -genkeypair -alias "zookeeperKey" -keystore "C:\IBM\i2analyze\i2-zookeeper-keystore.jks" -dname "CN=hostname" -keyalg RSA -storepass "password"
```

Important: Ensure that you enter values as follows:

- Set the value of CN to the host name of the server that hosts ZooKeeper.
- Set the location of the key database to the directory that contains the toolkit.
- Set the storepass attribute to be a secure password for the keystore.

Enter the key password for the ZooKeeper key or press the Return key to confirm that the key password is the same as keystore password.

b) Export the certificate from the ZooKeeper keystore.

For example, run the following command.

```
keytool -exportcert -alias "zookeeperKey" -keystore "C:\IBM\i2analyze\i2-zookeeper-keystore.jks" -file "C:\IBM\i2analyze\i2-zookeeper-certificate.cer" -storepass "password"
```

Enter the password that you set for the keystore in the previous step.

2. Create the ZooKeeper truststore and import the required certificates.

You must complete this for each server on which ZooKeeper is deployed.

a) If you are using a self-signed certificate, import the certificate that you exported from the ZooKeeper keystore in step 1b.

For example, run the following command:

```
keytool -importcert -alias "zookeeperKey" -keystore "C:\IBM\i2analyze\i2-zookeeper-truststore.jks" -file "C:\IBM\i2analyze\i2-zookeeper-certificate.cer" -storepass "password"
```

In response to the prompt, enter yes to trust the certificate.

b) Import the certificate that authenticates the certificates that are received from Solr. In this example, import the self-signed Solr `i2-solr-certificate.cer` certificate that you exported in [“Creating Solr keystores and certificates”](#) on page 247.

For example, run the following command:

```
keytool -importcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-zookeeper-truststore.jks" -file "C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
```

In response to the prompt, enter yes to trust the certificate.

c) Import the certificate that authenticates the certificates that are received from Liberty. In this example, import the self-signed Liberty `i2-liberty-certificate.cer` certificate that you exported in [“Creating the Liberty keystore and certificate”](#) on page 237.

For example, run the following command:

```
keytool -importcert -alias "libertyKey" -keystore "C:\IBM\i2analyze\i2-zookeeper-truststore.jks" -file "C:\IBM\i2analyze\i2-liberty-certificate.cer" -storepass "password"
```

In response to the prompt, enter yes to trust the certificate.

Configuring the SSL connections

To secure the connection between the i2 Analyze application server, Solr, and ZooKeeper, you must change the configuration of all three components. The i2 Analyze configuration must define the keystore and truststore for Solr and ZooKeeper.

- Ensure that you configured Liberty for SSL. For more information, see [“Configuring Liberty for SSL” on page 237](#).
- You must have the appropriate keystore set up for your Solr deployment. For more information, see [“Creating Solr keystores and certificates” on page 247](#).
- You must have the appropriate keystore set up for your ZooKeeper deployment. For more information, see [“Creating ZooKeeper keystores and certificates” on page 248](#).

Liberty uses a Java truststore to verify the certificate from the Solr and ZooKeeper servers, and so you must create or reuse a truststore on your i2 Analyze server to contain the trusted certificates for your Solr and ZooKeeper servers. You can use the same truststore that you created for Liberty to hold the certificate from the database management system. For more information, see [“Configuring i2 Analyze to connect to a database instance using SSL” on page 244](#).

1. Stop Liberty and ZooKeeper:

a) Stop Liberty.

To stop Liberty, run the following command on the Liberty server:

```
setup -t stopLiberty
```

Ensure that the Liberty instance is stopped, otherwise you encounter an error if you try to run the command when you complete the configuration changes.

b) Stop ZooKeeper.

To stop ZooKeeper, run the following command on every server where ZooKeeper is running:

```
setup -t stopZkHosts --hostname zookeeper.host-name
```

Here, *zookeeper.host-name* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the *host-name* attribute of a *<zkhost>* element in the *topology.xml* file.

Modify the configuration on the Liberty server.

2. Create the truststore and import the certificate that you exported from the Solr and ZooKeeper keystores into the truststore.

This truststore can be the truststore that you created for the certificate that you exported from the database management system. Alternatively, you create a new truststore if the file does not exist.

For example, run the following commands:


```
keytool -importcert -alias "solrKey" -keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks"
-file "C:\IBM\i2analyze\i2-solr-certificate.cer" -storepass "password"
keytool -importcert -alias "zookeeperKey" -keystore "C:\IBM\i2analyze\i2-liberty-
truststore.jks" -file "C:\IBM\i2analyze\i2-zookeeper-certificate.cer" -storepass "password"
```

In response to the prompt, enter yes to trust the certificate.

Important: Ensure that you enter values as follows:

- Enter unique aliases.
- Set the location of the keystore to the directory that contains the toolkit.
- Assign secure passwords.

Add the ZooKeeper certificate to the Solr truststore on the Solr server.

3. Import the certificate that you exported from the ZooKeeper keystore into the Solr truststore.

For example, run the following command:

```
keytool -importcert -alias "zookeeperKey" -keystore "C:\IBM\i2analyze\i2-solr-truststore.jks"
-file "C:\IBM\i2analyze\i2-zookeeper-certificate.cer" -storepass "password"
```

In response to the prompt, enter yes to trust the certificate.

Important: Ensure that you enter values as follows:

- Enter a unique alias.
- Set the location of the keystore to the directory that contains the toolkit.
- Assign a secure password.

4. Modify the `topology.xml` file to specify SSL for its Solr connection.

a) In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

b) In the `<solr-cluster>` element for the Solr cluster that you want to connect to with SSL, add the `secure-connection` attribute with the value of `true`.

For example, add the attribute as highlighted in the following code:

```
<solr-cluster id="is_cluster" zookeeper-id="zoo" secure-connection="true">
```

c) Add the `key-store` and `trust-store` attributes to either the `<solr-cluster>` or the `<solr-node>` element.

Add the attribute values as defined:

key-store

The path to the Solr keystore. For more information, see [“Creating Solr keystores and certificates”](#) on page 247. Reference step 1a.

trust-store

The path to the Solr truststore. For more information, see [“Creating Solr keystores and certificates”](#) on page 247. Reference step 2a.

For example, add the attribute as highlighted in the `<solr-clusters>` element:

```
<solr-cluster id="is_cluster" zookeeper-id="zoo" secure-connection="true"
key-store="C:\IBM\i2analyze\i2-solr-keystore.jks" trust-store="C:\IBM\i2analyze\i2-solr-
truststore.jks">
```

Or add the attribute as highlighted in the <solr-node> element:

```
<solr-node memory="2g" data-dir="C:\IBM\i2analyze\data\solr" host-name="hostname"
id="node1" port-number="8983"
  key-store="C:\IBM\i2analyze\i2-solr-keystore.jks" trust-store="C:\IBM\i2analyze\i2-solr-
truststore.jks">
```

Note: The host-name attribute value must match the common name that is associated with the certificate for Solr. For more information, see [“Creating Solr keystores and certificates” on page 247](#). Reference step 1a.

5. Modify the topology.xml file to specify SSL for its ZooKeeper connection.

- a) In an XML editor, open the toolkit\configuration\environment\topology.xml file.
- b) In the <zookeeper> element for the ZooKeeper host that you want to connect to with SSL, add the secure-connection attribute with the value of true.

For example, add the attribute as highlighted in the following code:

```
<zookeeper id="zoo" secure-connection="true">
```

- c) Add the key-store and trust-store attributes to either the <zookeeper> or to the <zookeeper> element.

Add the attribute values as defined:

key-store

The path to the ZooKeeper keystore. For more information, see [“Creating ZooKeeper keystores and certificates” on page 248](#). Reference step 1a.

trust-store

The path to the ZooKeeper truststore. For more information, see [“Creating ZooKeeper keystores and certificates” on page 248](#). Reference step 2a.

- For example, add the attribute as highlighted in the <zookeeper> element:

```
<zookeeper id="zoo" secure-connection="true" key-store="C:\IBM\i2analyze\i2-zookeeper-
keystore.jks" trust-store="C:\IBM\i2analyze\i2-zookeeper-truststore.jks">
```

- For example, add the attribute as highlighted in the <zookeeper> element:

```
<zookeeper quorum-port-number="10483" leader-port-number="10983" data-
dir="C:\IBM\i2analyze\data\zookeeper" host-name="hostname" id="1" port-number=" 9983"
  key-store="C:\IBM\i2analyze\i2-zookeeper-keystore.jks" trust-store="C:\IBM\i2analyze\i2-
zookeeper-truststore.jks">
```

Note: The host-name attribute value must match the common name that is associated with the certificate for ZooKeeper. For more information, see [“Creating ZooKeeper keystores and certificates” on page 248](#). Reference step 1a.

6. Modify the topology.xml file to add the Liberty truststore.

Add a child <key-store> element. For your keystore, specify the type as trust-store and file as the full path to your truststore.

For example, add the element as highlighted in the following code:

```

<application http-server-host="true" name="opal-server"
  host-name="hostname" secure-connection="true">
  ...
  <key-stores>
    <key-store type="key-store"
      file="C:\IBM\i2analyze\i2-liberty-keystore.jks"/>
    <key-store type="trust-store"
      file="C:\IBM\i2analyze\i2-liberty-truststore.jks"/>
  </key-stores>
  ...
</application>

```

7. Specify the truststore and keystore passwords in the credentials file.

- a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
- b) Enter the passwords for the Solr keystore and truststore that you specified in the topology file.

```

solr.truststore.password=password
solr.keystore.password=password

```

- c) Enter the passwords for the ZooKeeper keystore and truststore that you specified in the topology file.

```

zookeeper.truststore.password=password
zookeeper.keystore.password=password

```

- d) Enter the password for the Liberty truststore that you specified in the topology file.

```

ssl.truststore.password=password

```

8. Copy the `toolkit\configuration` from the Liberty server, to the `toolkit` directory of the deployment toolkit on each server in your environment.

Update the application with your configuration changes. Run the following commands from the `toolkit\scripts` directory on the Liberty server.

9. Redeploy Liberty to update the application:

```

setup -t deployLiberty

```

10. Recreate the ZooKeeper host on each server where your ZooKeeper hosts are located:

```

setup -t createZkHosts --hostname zookeeper.host-name

```

Where *zookeeper.host-name* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the `topology.xml` file.

11. Start ZooKeeper.

To start ZooKeeper, run the following command on every server where your ZooKeeper hosts are located:

```

setup -t startZkHosts --hostname zookeeper.host-name

```

Where *zookeeper.host-name* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the *host-name* attribute of a *<zghost>* element in the *topology.xml* file.

12. Upload the new Solr configuration to ZooKeeper:

```
setup -t createAndUploadSolrConfig --hostname liberty.hostname
```

Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the *host-name* attribute of the *<application>* element in the *topology.xml* file.

13. Restart the Solr nodes.

To restart the Solr nodes, run the following command on every server where Solr is running:

```
setup -t restartSolrNodes --hostname solr.host-name
```

Where *solr.host-name* is the host name of the Solr server where you are running the command, and matches the value for the *host-name* attribute of a *<solr-node>* element in the *topology.xml* file.

14. Start Liberty.

To start Liberty, run the following command on each Liberty server:

```
setup -t startLiberty
```

Connect to your Information Store. For more information, see [Connecting IBM i2 Analyst's Notebook Premium to IBM i2 Analyze](#).

Search for data to ensure that the Solr connection is working.

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although IBM does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate, and maintain applications that can be trusted. In particular, see https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks and https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

SANS Institute

The System-Admin, Audit, Network, and Security Institute is the largest source for [information security training and security certification](#) in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the internet's early warning system - the [Internet Storm Center](#). In particular, see <https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines>

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

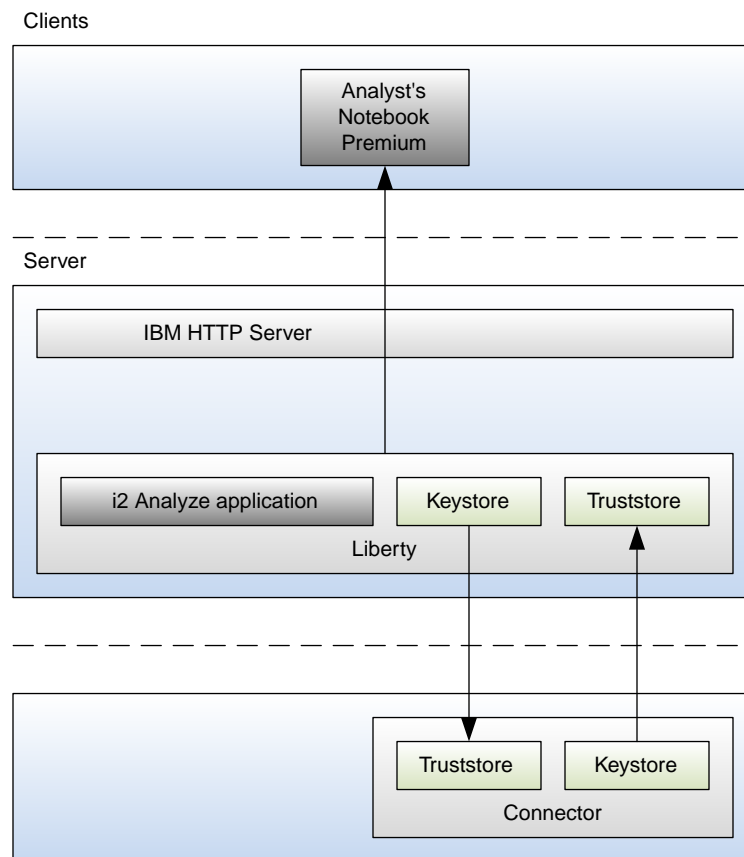
In particular, see <http://cwe.mitre.org/top25/index.html#CWE-307>.

Client authenticated Secure Sockets Layer with the i2 Connect gateway

To secure the connection between Liberty and any connectors for the i2 Connect gateway, you must configure Liberty and your connectors to use SSL. If you are using SSL, i2 Analyze enforces client-authenticated communication with a connector.

In a production deployment you should configure i2 Analyze to connect to your connector using client-authenticated SSL communication. To do so, your connector and i2 Analyze must trust the certificates that they receive during the SSL handshake process. In a production environment, the certificates must be signed by a trusted certificate authority. For more information about client authenticated SSL, see [Client-authenticated TLS handshake](#).

The following diagram shows the keystores and truststores that are required for Liberty and the connector.



The Liberty server requires a keystore file and a truststore file. Your connector can use any implementation for its keystore and truststore. The certificates in each truststore must trust the certificates received from the corresponding keystore.

The certificates that are required are as follows, where certificate authority (CA) X issues the certificates to the connector (the server) and Liberty (the client):

The connector requires:

- In its keystore:
 - The personal certificate issued to the connector by CA X
 - The connector's private key
- In the truststore:
 - The CA certificate for CA X

Liberty requires:

- In its keystore:
 - The personal certificate issued to Liberty by CA X
 - Liberty's private key
- In its truststore:
 - The CA certificate for CA X

After you have created and populated the keystores and truststores for the connector and Liberty, you must configure Liberty to use SSL to communicate with any connectors.

The following steps explain the process of updating the i2 Analyze configuration with the location of a keystore and truststore to use, and the passwords that are used to access the certificates that are contained within them.

To configure the example-connector to use client-authenticated SSL, and for examples of how to create keystores, truststores, and certificates for Liberty, follow the instructions in [“Securing the example connector” on page 257](#).

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

- a) Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.

For your keystore, specify the `type` as `key-store`, and `file` as the full path to your keystore. For your truststore, specify the `type` as `trust-store`, and `file` as the full path to your truststore.

For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true"
  name="opal-server" host-name="hostname">
  ...
  <key-stores>
    <key-store type="key-store"
      file="C:/IBM/i2analyze/i2-liberty-keystore.jks"/>
    <key-store type="trust-store"
      file="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
  </key-stores>
  ...
</application>
```

- b) Update the `base-url` attribute of any connectors using SSL to use the HTTPS protocol.

For example:

```
<connectors>
  <connector id="example-connector" name="Example"
    base-url="https://localhost:3700/" />
</connector>
```

Note: Ensure that the hostname that is used in the base URL matches the common name on the certificate of the connector.

2. Specify the keystore and truststore passwords in the credentials file.
 - a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
 - b) Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

```
ssl.keystore.password=password
ssl.truststore.password=password
```

Redeploy i2 Analyze to update the application with your changes.

3. In a command prompt, navigate to the `toolkit\scripts` directory.
4. Stop Liberty:

```
setup -t stopLiberty
```

5. Update the i2 Analyze application:

```
setup -t deployLiberty
```

6. Start Liberty:

```
setup -t startLiberty
```

You can create your own connectors to use with the deployment of i2 Analyze, when you create your own connector you can implement security that conforms to the security required by the i2 Connect gateway. For more information about creating your own connectors, see [IBM i2 Analyze and i2 Connect](#).

When you use a connector configured for SSL communication, you should not see any warnings displayed in Analyst's Notebook Premium.

Securing the example connector

In a production deployment, you must configure your connectors with client-authenticated SSL communication. You can configure the example connector to communicate with i2 Analyze by using client-authenticated SSL communication.

You must configure your connectors to use client-authenticated SSL communication with i2 Analyze. For more information about configuring security between connectors and i2 Analyze, see ["Client authenticated Secure Sockets Layer with the i2 Connect gateway"](#) on page 255.

Configure secure communication between Liberty and the example connector that is provided with i2 Analyze. The following steps demonstrate how to create the keystores, truststores, and certificates, then configure i2 Analyze and the example connector to use them.

When you create your own connector, you can implement the security implementation in the same way as used in the example connector, or a different method depending on your requirements.

In a production deployment, you must use a certificate that has been signed by a trusted certificate authority, which the connector can verify. The following steps use a self-signed certificate to demonstrate the functionality.

Create a keystore for Liberty that contains a signed certificate that is used to authenticate with the connector.

1. Create a keystore and a self-signed certificate for Liberty by using the Java keytool utility.

For more information about the Java keytool utility, see [keytool - Key and Certificate Management Tool](#).

- a) Open a command prompt and navigate to the `i2analyze\deploy\java\bin` directory.
- b) Create the keystore and certificate by running the following command:

```
keytool -genkeypair -alias "libertyKey"  
-keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks"  
-dname "CN=hostname" -keyalg RSA -storepass "libertyKeyStorePassword"
```

Important: Ensure that you enter values as follows:

- Set the location of the keystore to the directory that contains the toolkit.
- Set the value of CN to the hostname of the server that hosts Liberty, as defined in the `topology.xml` file.

The password that is specified is used to access the keystore.

- c) The example connector implementation requires the Liberty certificate in a base64 encoded format. Export the certificate in base64 encoding by running the following command:

```
keytool -exportcert -alias "libertyKey"  
-keystore "C:\IBM\i2analyze\i2-liberty-keystore.jks"  
-file "C:\IBM\i2analyze\i2-liberty-certificate.cer"  
-rfc -storepass "libertyKeyStorePassword"
```

Create a keystore for the connector that contains a signed certificate that is used to authenticate with Liberty.

2. Create a keystore and certificate for the connector by using the IBM Key Management Utility command-line interface.

For more information about the IBM Key Management Utility, see [Key Management Utility command-line interface \(gskcmd\) syntax](#).

- a) In a command prompt, navigate to the `bin` directory of your IBM HTTP Server installation. For example: `IBM\HTTPServer\bin` or `IBM\HTTPServer\IHS\bin`.
- b) Create a keystore in PKCS12 format. To create the keystore, run the following command:

```
gskcapicmd -keydb -create -type pkcs12  
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-  
keystore.p12"  
-pw "connectorKeyStorePassword"
```

The password that is specified is used to access the keystore.

- c) Create a self-signed certificate by running the following command:


```
gskcapicmd -cert -create
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-keystore.p12"
-label "connectorKey" -dn "CN=hostname" -pw "connectorKeyStorePassword"
```

Important: Set the value of CN to the hostname of the server that hosts the connector, as defined in the topology.xml file. By default in the daod-opa1 example configuration, the value is localhost.

3. The example connector implementation requires the private key in the PEM format, and the certificate for the connector.

- a) Export the personal certificate and private key into a PKCS12 file by running the following command:

```
gskcapicmd -cert -export
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-keystore.p12"
-pw "connectorKeyStorePassword" -label "connectorKey" -type pkcs12
-target "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-key.p12"
-target_pw "connectorKeyPassword" -target_type pkcs12
```

- b) Convert the private key from the PKCS12 format to the PEM format. You can use OpenSSL to do this. For more information about OpenSSL see <https://www.openssl.org/source/>. If you are using OpenSSL, you can run the following command:

```
openssl pkcs12 -in C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-key.p12
-out C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-key.pem
-nocerts -nodes
```

When you are prompted for a password, provide the password that you specified when you created the example-connector-key.p12 PKCS12 file. In this example, enter connectorKeyPassword.

- c) Extract the certificate for the connector by running the following command:

```
gskcapicmd -cert -extract
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-keystore.p12"
-label "connectorKey"
-target "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-certificate.cer"
-pw "connectorKeyStorePassword"
```

Create a truststore for Liberty that contains the certificate that is used to trust the certificate that it receives from the connector.

4. Create the Liberty truststore and populate it with the connector's certificate by using the Java keytool by running the following command:

```
keytool -importcert -alias "exampleconnector"  
-keystore "C:\IBM\i2analyze\i2-liberty-truststore.jks"  
-file "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-  
certificate.cer"  
-storepass "libertyTrustStorePassword"
```

In response to the prompt, enter yes to trust the certificate.

5. Populate the connector keystore with Liberty's certificate by using the IBM Key Management Utility by running the following command:

```
gskcapicmd -cert -add -type pkcs12  
-db "C:\IBM\i2analyze\toolkit\examples\connectors\example-connector\example-connector-  
keystore.p12"  
-pw "connectorKeyStorePassword" -label "libertyKey" -trust enable  
-file "C:\IBM\i2analyze\i2-liberty-certificate.cer"
```

6. The example connector implementation requires the certificate that is used to trust Liberty to also be present in the example-connector directory. Copy the C:\IBM\i2analyze\i2-liberty-certificate.cer file to the C:\IBM\i2analyze\toolkit\examples\connectors\example-connector directory.
7. If you are using the IBM HTTP Server, start or restart it.

The certificates are now in place to enable client authentication SSL between Liberty and the connector.

8. Configure the example connector to reference the certificates that you created, and the hostname of the gateway. Using a text editor, modify the security-config.json file with the following values.

https

Set to true to use the HTTPS protocol to connect to the connector.

keyFileName

The file name for the private key of the connector in PEM format. For this example, example-connector-key.pem.

keyPassphrase

The password that is required to access the key file specified in keyFileName. For this example, connectorKeyPassword.

certificateFileName

The file name for the certificate of the connector. For this example, example-connector-certificate.cer.

certificateAuthorityFileName

The file name of the certificate that enables trust of the certificate that is received from Liberty. For this example, i2-liberty-certificate.cer.

gatewayCN

The common name of the gateway. This must be the value of the common name in the certificate the connector receives from Liberty. You specified the value of the CN in the certificate in step 1.

Configure Liberty to use the keystore and truststore that you created.

9. In an XML editor, open the toolkit\configuration\environment\topology.xml file.

- a) Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.

For your keystore, specify the type as `key-store`, and `file` as the full path to your keystore. For your truststore, specify the type as `trust-store`, and `file` as the full path to your truststore.

For example, add the attribute as highlighted in the following code:

```
<application http-server-host="true"
  name="opal-server" host-name="hostname">
  ...
  <key-stores>
    <key-store type="key-store"
      file="C:/IBM/i2analyze/i2-liberty-keystore.jks"/>
    <key-store type="trust-store"
      file="C:/IBM/i2analyze/i2-liberty-truststore.jks"/>
  </key-stores>
  ...
</application>
```

- b) Update the `base-url` attribute of any connectors using SSL to use the HTTPS protocol. For example:

```
<connectors>
  <connector id="example-connector" name="Example"
    base-url="https://localhost:3700/" />
</connector>
```

Note: Ensure that the hostname that is used in the base URL matches the common name on the certificate of the connector.

10. Specify the keystore and truststore passwords in the credentials file.

- a) In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
b) Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

```
ssl.truststore.password=libertyTrustStorePassword
ssl.keystore.password=libertyKeyStorePassword
```

11. In a command prompt, navigate to the `toolkit\scripts` directory.

12. Update the i2 Analyze application:

```
setup -t deployLiberty
```

13. Start the example connector.

- a) In a new command prompt, navigate to the `toolkit\examples\connectors\example-connector` directory.
b) To start the Node.js server, run the following command:

```
npm start
```

Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

14. Start Liberty:

```
setup -t startLiberty
```

15. If you are using the IBM HTTP Server, start or restart it.

Use Analyst's Notebook Premium to connect to your deployment. For more information, see [Connecting IBM i2 Analyst's Notebook Premium to IBM i2 Analyze](#).

Now that the example connector is configured for SSL, no warnings are displayed in Analyst's Notebook Premium.

Adding more data sources

You can extend your system to use different sources of data by adding the i2 Connect gateway, and configuring your deployment to identify the data source and searching to use. You can add the i2 Connect gateway to an existing deployment of i2 Analyze.

You must have a deployment of i2 Analyze. When you add the i2 Connect gateway to your deployment, the configuration for an example connector is included. To use the example connector, you must download and install Node.js to host the example connector. Download Node.js for your operating system from: <https://nodejs.org/en/download/>. You can install Node.js with the default settings.

To use a custom connector that is tailored to your deployment, you either need the details of a connector that is provided to you, or to create a connector. For more information about how to create a custom connector, see [Connecting to external data sources](#).

The i2 Connect gateway enables analysts to search for and retrieve data from external data sources, and then analyze the results on a chart in Analyst's Notebook Premium. To use the i2 Connect gateway, you must obtain or create a custom connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and an example connector.

1. Run the addI2Connect task to add the i2 Connect gateway to your deployment. In a command prompt, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -t addI2Connect
```

2. Decide which connector you would like to add to your deployment:

- If you want to use the example connector, complete the step to start the example connector in [Deploying i2 Analyze with the Information Store and the i2 Connect gateway](#).
- If you would like to add a custom connector, replace the following elements from the `configuration\environment\topology.xml` file with elements specific to the connector you want to use:
 - The `<connector>` element with the ID `example-connector`
 - The `connector-id` element with the value `example-connector`

Note: For more information about making connector-specific changes to the topology file, see [Adding a connector to the topology](#).

3. Redeploy i2 Analyze.

```
setup -t deploy
```

4. Start i2 Analyze.

```
setup -t start
```

5. Start, or restart, the HTTP server that hosts the reverse proxy.

When you start i2 Analyze, the URI that users must specify in Analyst's Notebook Premium is displayed in the console. For example, This application is configured for access on `http://host_name/opa1`.

Install Analyst's Notebook Premium and connect to your deployment. For more information, see [Installing IBM i2 Analyst's Notebook Premium](#) and [Connecting IBM i2 Analyst's Notebook Premium to IBM i2 Analyze](#).

Note: The example connector does not use client authenticated SSL communication between i2 Analyze and any connectors, so a warning is displayed in Analyst's Notebook Premium. For more information about configuring client-authenticated SSL, see [Client authenticated Secure Sockets Layer with the i2 Connect gateway](#).

Back up and restore

An effective backup strategy considers the key components in your deployment and assess the impact of creating replica versions versus the cost of replacing those components if the system fails. All production deployments of i2 Analyze should have a backup and restore strategy in place.

The following components of an i2 Analyze deployment should be backed up to provide a comprehensive backup strategy:

- The configuration
- The Solr search index and ZooKeeper configuration (If you have an Information Store or Chart Store)
- The database (If you have an Information Store or Chart Store)

When you create a backup strategy, consider the following items:

- The amount of time, if any, that is acceptable for a system to be offline to create backups.
- The amount of time that is acceptable for a system to be offline while recovery is in progress.
- The amount of time between backups, for each component.

In a deployment of i2 Analyze that contains a database, the database is the source of truth about the data contained in a deployment.

This means that the Solr search index must be backed up before the database is and that the database must be restored before the Solr search index. If any data in the Information Store was modified after the Solr search index was backed up, the changes are updated in the index after the restoring.

The i2 Analyze deployment toolkit contains toolkit tasks that you can use to back up and restore each component of i2 Analyze. You can choose to use the toolkit tasks to back up all components, or you can use the toolkit tasks to back up a subset and use the native back up and restore procedure provided by other components. For example, if your organization has existing processes for backing up and restoring databases but not a Solr search index, you can use the existing process to back up the database in a deployment and the provided toolkit task to back up the Solr search index.

Backing up a deployment

In a production deployment of i2 Analyze, use a tested backup and restore procedure to ensure that you can recover from failures. The various components of i2 Analyze must be backed up for complete backup coverage.

About this task

The following steps describe the process of backing up a deployment of i2 Analyze by using the provided toolkit tasks. You do not need to back up every component by using the toolkit tasks. For example, you can use the toolkit task to back up Solr and use existing processes to back up the database. Regardless of the mechanism that you use to back up the components, you must take the Solr backup before the database backup.

The configuration and Solr backups can be completed while the deployment is running. For more information, see:

- [Back up and restore Solr](#)
- [Back up and restore the configuration](#)

When you use the toolkit task to back up the database, you must stop the deployment first. To perform an online backup of the database, see the documentation for your database management system. For more information about backing up the database using the toolkit, see: [Back up and restore the database](#).

In a deployment that contains the i2 Connect gateway only, you only need to backup the i2 Analyze and Liberty configuration.

Run each toolkit command from a Liberty server in your deployment.

Procedure

1. In the `environment.properties` file, specify the locations where the toolkit creates the backup files.

`backup.config.location.dir`

The location where the i2 Analyze configuration backups are created and restored from. This location must exist on the Liberty server where you run the `backupConfiguration` command.

`backup.solr.location.dir`

The location where the Solr index and ZooKeeper configuration backups are created and restored from. This location must be accessible by every Solr node in your deployment.

`backup.db.location.dir`

The location where the database backups are created and restored from. The user that is specified in the `credentials.properties` file for your database management system must have write permissions to this location. This location must be accessible on the database server.

For example:

```
backup.config.location.dir=C:/backup/configuration
backup.solr.location.dir=D:/network-drive/solr/backup
backup.db.location.dir=D:/backup/database
```

2. To back up the i2 Analyze and Liberty configuration, run the following command from a Liberty server.

In a deployment with high availability, run this command on each Liberty server.

```
setup -t backupConfiguration
```

3. To back up the Solr search index, run the following command from a Liberty server:

```
setup -t backupSolr
```

4. To back up the database in your deployment by using the toolkit task, you must stop the application first, and ensure that there are no other connections to the database, then run the

backup command. In a deployment that contains multiple Liberty servers, you must run the stopLiberty command on every Liberty server.

```
setup -t stopLiberty
setup -t backupDatabases
setup -t startLiberty
```

Restoring a deployment to a point in time

Use the backups of the components of i2 Analyze to restore the system to a previous point in time. In a production deployment of i2 Analyze, use a tested backup and restore procedure to ensure that you can recover from failures.

About this task

The following steps describe the process of restoring a deployment of i2 Analyze by using the provided toolkit tasks. You do not need to restore every component by using the toolkit tasks. For example, you can use existing processes to restore the database and use the toolkit task to restore Solr. Regardless of the mechanism that you use to restore the components, you must restore the database before you restore the Solr index.

To deploy to a previous point in time, your deployment must be functional. The procedure assumes that Solr, ZooKeeper, and the database management system are running. As part of the process, you must stop the Liberty servers in the deployment. To recover a deployment after a disaster, see [Recovering from a disaster](#).

In a deployment that contains the i2 Connect gateway only, you only need to restore the i2 Analyze and Liberty configuration.

For more information about the restore toolkit tasks, see:

- [Back up and restore the database](#)
- [Back up and restore Solr](#)
- [Back up and restore the configuration](#)

Run each toolkit command from a Liberty server in your deployment.

Procedure

1. Use the validateBackups to ensure that the timestamps of the backups you are restoring are compatible.

This task compares the timestamps of the backups created by the toolkit to ensure that the configuration backup is the earliest, and that the Solr backup precedes the database backup.

```
setup -t validateBackups -p configTimestamp=<timestamp> -p solrTimestamp=<timestamp> -p
dbTimestamp=<timestamp>
```

Where configTimestamp is the timestamp of a configuration backup, solrTimestamp is the timestamp of a Solr backup, and dbTimestamp is the timestamp of a database backup. You do not have to provide all of the timestamps to run the toolkit task.

If the backups are compatible, continue with the restore process.

2. Before you restore the deployment, stop Liberty. In a deployment that contains multiple Liberty servers, you must run the `stopLiberty` command on every Liberty server.

```
setup -t stopLiberty
```

3. To restore the database in your deployment, run the following command from a Liberty server:

```
setup -t restoreDatabases -p timestamp=<timestamp>
```

The database backup is identified in the location from the `backup.db.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup in the location.

For example: `setup -t restoreDatabases -p timestamp=20210420145332`.

4. To restore the Solr search index, run the following command from a Liberty server:

```
setup -t restoreSolr -p timestamp=<timestamp> --hostname <liberty.hostname>
```

Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

The Solr backup is identified in the location from the `backup.solr.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup directory in the location.

For example: `setup -t restoreSolr -p timestamp=20210420145332 --hostname <liberty.hostname>`.

5. To restore the i2 Analyze and Liberty configuration, run the following command from a Liberty server:

```
setup -t restoreConfiguration -p timestamp=<timestamp>
```

The configuration backup is identified in the location from the `backup.config.location.dir` setting by using the timestamp that was specified in the command.

The timestamp value in the command must match the timestamp of a backup directory in the location.

In a deployment with high availability, run this command on each Liberty server.

For example: `setup -t restoreConfiguration -p timestamp=20210420145332`.

6. Start Liberty.

In a deployment that contains multiple Liberty servers, you must run the `startLiberty` command on every Liberty server.

```
setup -t startLiberty
```

Result

Your deployment of i2 Analyze is restored to the point in time of the backups that you specified.

Recovering from a disaster

You can use the backups of the components of i2 Analyze to restore the system, or parts of a the system, after a disaster. In a production deployment of i2 Analyze, use a tested backup and restore procedure to ensure that you can recover from failures.

About this task

The following steps describe the process of restoring a deployment of i2 Analyze by using the provided toolkit tasks. You do not need to restore every component by using the toolkit tasks. For example, you can use existing processes to restore the database and use the toolkit task to restore Solr. Regardless of the mechanism that you use to restore the components, you must restore the components in the same order as the following steps.

For more information about the restore toolkit tasks, see:

- [Back up and restore the database](#)
- [Back up and restore Solr](#)
- [Back up and restore the configuration](#)

In a deployment that contains the i2 Connect gateway only, you only need to restore the i2 Analyze and Liberty configuration.

To recover from a disaster, you might need to reinstall the i2 Analyze deployment toolkit and install the components of i2 Analyze on the failed servers. The following process assumes that you are recreating the servers for each component in your deployment.

Procedure

1. Create the environment to restore your deployment in to. If your servers were lost due to a disaster, you might need to recreate them with the required prerequisites. For more information about the servers that are used in a deployment of i2 Analyze, see [Deployment topologies](#).
2. Use the `validateBackups` to ensure that the timestamps of the backups you are restoring are compatible.
This task compares the timestamps of the backups created by the toolkit to ensure that the configuration backup is the earliest, and that the Solr backup precedes the database backup.

```
setup -t validateBackups -p configTimestamp=<timestamp> -p solrTimestamp=<timestamp> -p dbTimestamp=<timestamp>
```

Where `configTimestamp` is the timestamp of a configuration backup, `solrTimestamp` is the timestamp of a Solr backup, and `dbTimestamp` is the timestamp of a database backup. You do not have to provide all of the timestamps to run the toolkit task.

If the backups are compatible, continue with the restore process.

3. Restoring to new Liberty servers.
 - a. Copy the `i2a-config-<timestamp>\toolkit-config` directory from your backup to the `toolkit\configuration` directory of the deployment toolkit on each server in your environment.
You might have a different configuration backup for each Liberty server in your deployment, ensure that you copy the correct backup for each Liberty server.
4. Create a new deployment to restore in to. To create a new deployment, follow the instructions to install, deploy, and start the components in [Deploying i2 Analyze on multiple servers](#).

By starting the components, you ensure that your environment is deployed successfully. You can now restore your environment from your backups.

5. Before you restore the deployment, stop Liberty. In a deployment that contains multiple Liberty servers, you must run the `stopLiberty` command on every Liberty server.

```
setup -t stopLiberty
```

6. To restore the database in your deployment, run the following command from a Liberty server:

```
setup -t restoreDatabases -p timestamp=<timestamp>
```

The database backup is identified in the location from the `backup.db.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup in the location.

For example: `setup -t restoreDatabases -p timestamp=20210420145332`.

7. To restore the Solr search index, run the following command from a Liberty server:

```
setup -t restoreSolr -p timestamp=<timestamp> --hostname <liberty.hostname>
```

Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

The Solr backup is identified in the location from the `backup.solr.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup directory in the location.

For example: `setup -t restoreSolr -p timestamp=20210420145332 --hostname <liberty.hostname>` .

8. Deploying to new Liberty servers.

- a. To restore the i2 Analyze and Liberty configuration, run the following command from a Liberty server:

```
setup -t restoreConfiguration -p timestamp=<timestamp>
```

- The configuration backup is identified in the location from the `backup.config.location.dir` setting by using the timestamp that was specified in the command.
- The timestamp value in the command must match the timestamp of a backup directory in the location.
- In a deployment that contains multiple Liberty servers, run the `restoreConfiguration` command on each Liberty server.

For example: `setup -t restoreConfiguration -p timestamp=20210420145332`.

- b. Start Liberty.

In a deployment that contains multiple Liberty servers, you must run the `startLiberty` command on each Liberty server.

```
setup -t startLiberty
```

Result

Your deployment of i2 Analyze is restored to new servers after a disaster.

Back up and restore the configuration

The i2 Analyze and Liberty configuration contains the files that define the functionality of the deployment, and how users connect to the deployment. To restore the deployment after a failure, create a backup of the configuration.

Intro

- The configuration directory and some files from the Liberty deployment are backed up. These files enable you to recreate a deployment from the backup files in the same configuration.
- The system can be running when you back up the configuration. The system must be stopped when you restore the configuration.
- In a distributed deployment of i2 Analyze, the configuration directory should be identical on each server. Therefore, you only need to back up the configuration from the Liberty server.

Backup location

In the `environment.properties` file, the `backup.config.location.dir` setting is the location where the configuration backups are created and restored from when you use the toolkit tasks. This location must exist on the Liberty server where you run the toolkit tasks.

Toolkit tasks

- `backupConfiguration`
 - The `backupConfiguration` toolkit task backs up the i2 Analyze and Liberty configuration to the `backup.config.location.dir` location.
For example:

```
setup -t backupConfiguration
```

- `restoreConfiguration`
 - The `restoreConfiguration` toolkit task restores the i2 Analyze and Liberty configuration from a backup. The `restoreConfiguration` also deploys Liberty. You must pass the `timestamp` parameter to the toolkit task with a timestamp for a backup in the `backup.config.location.dir` location.
For example:

```
setup -t restoreConfiguration -p timestamp=<timestamp>
```

If you are recovering from a disaster scenario where you must create new Liberty servers, you must copy the i2 Analyze deployment toolkit configuration from the backup to your new Liberty servers before you run the `restoreConfiguration` task.

Backup files

When the `backupConfiguration` task is run, a directory is created named `i2a-config-<timestamp>` where `<timestamp>` is the timestamp of when the toolkit task was run. The directory contains the

files of the i2 Analyze and Liberty configuration and a file that shows the version of i2 Analyze. For example:

```
- i2a-config-20210420145332
  - version.txt
  - liberty-app-config
  - liberty-shared-config
  - toolkit-config
```

The following directories are included in the backup:

- toolkit\configuration
- wlp\usr\shared\configs
- wlp\usr\servers\opal-server

Back up and restore Solr

When you use the `backupSolr` and `restoreSolr` toolkit tasks to back up and restore the Solr index and ZooKeeper configuration, a call to the Solr REST API is made. The toolkit tasks create and restore a full backup of the non-transient Solr collections.

Intro

- The non-transient Solr collections are backed up and restored by using the provided toolkit tasks. The following collection types are non-transient:
 - main_index
 - match_index
 - chart_index
- You must take the Solr backup before a database backup, and you must restore Solr after you restore the database.

Backup location

In the `environment.properties` file, the `backup.solr.location.dir` setting is the location where the Solr index and ZooKeeper configuration backups are created and restored from. This location must be accessible by every Solr node in your deployment.

Toolkit tasks

- `backupSolr`
The `backupSolr` toolkit task backs up the Solr index and ZooKeeper configuration to the `backup.solr.location.dir` location.
For example:

```
setup -t backupSolr
```

- `restoreSolr`
The `restoreSolr` toolkit task restores the Solr index and ZooKeeper configuration from a backup. You must pass the `timestamp` parameter to the toolkit task with a value for a backup in the `backup.solr.location.dir` location.

You must also specify the `hostname` parameter, with the hostname of the Liberty server where you are running the command, and that matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file. For example:

```
setup -t restoreSolr -p timestamp=<timestamp> --hostname <liberty.hostname>
```

Backup files

When the `backupSolr` task is run, a directory is created named `i2a-solr-<collection_name>-<timestamp>` where `<timestamp>` is the timestamp of when the toolkit task was run and `<collection_name>` the name of the Solr collection backed up. A directory is created for each non-transient Solr collection, with each collection containing a back up. For example:

```
- i2a-solr-chart_index-20210420145332
  - backup.properties
  - snapshot.shard1
  - snapshot.shard2
  - zk_backup
- i2a-solr-main_index-20210420145332
  - backup.properties
  - snapshot.shard1
  - snapshot.shard2
  - zk_backup
- i2a-solr-match_index1-20210420145332
  - backup.properties
  - snapshot.shard1
  - snapshot.shard2
  - zk_backup
- i2a-solr-match_index2-20210420145332
  - backup.properties
  - snapshot.shard1
  - snapshot.shard2
  - zk_backup
```

Back up and restore the database

When you use the `backupDatabases` and `restoreDatabases` toolkit tasks to back up and restore the database in your deployment, a call to the database management system is made. The toolkit tasks create and restore a full backup of the database.

Intro

- The `backupDatabases` toolkit task completes a full backup of the database. The application must be stopped before you can back up the database and there must be no active connections to the database.
 - Refer to your database management system documentation to learn how to ensure that there are no connections to your database before you create the backup.
- The toolkit task is aimed for smaller deployments where the system can be offline while the database backup is completed. If you have a larger deployment or don't want the system to be offline at all, you can back up the database manually. To back up and restore the database manually refer to the documentation for your database management system:

- For Db2, see [Developing a backup and recovery strategy](#)
- For SQL Server, see [Backup Overview](#)
- If you back up the database manually, you can still use the toolkit tasks to back up the i2 Analyze configuration and the Solr index.
- When you back up a deployment of i2 Analyze, the database must be backed up after the Solr index. When you restore the database, you must restore the database before the Solr index.

Backup location

In the `environment.properties` file, the `backup.db.location.dir` setting is the location where the database backup are created and restored from. This location must be accessible from the database server. The user that is specified in the `credentials.properties` file for your database management system must have write permissions to this location.

Toolkit tasks

- `backupDatabases`
The `backupDatabases` toolkit task backs up the database to the `backup.db.location.dir` location. For example:

```
setup -t backupDatabases
```

- `restoreDatabases`
The `restoreDatabases` toolkit task restores the database from a backup. You must pass the `timestamp` parameter to the toolkit task with a value for a backup in the `backup.db.location.dir` location. For example:

```
setup -t restoreDatabases -p timestamp=<timestamp>
```

Backup files

A backup file is created in the location specified in `backup.db.location.dir`. The file name includes a timestamp of when the backup is created. For example:

- On Db2: `ISTORE.0.db2inst1.DBPART000.20210604144506.001`
- On SQL Server: `ISTORE.20210604152638.bak`

You can verify the backup file was correctly created by running the following commands on your database server:

- On Db2: `db2ckbcp <backup file>`
- On SQL Server:

```
RESTORE VERIFYONLY FROM DISK = "<backup file>";
GO
```

Administering high availability

When your system is deployed in an environment that is designed for high availability or disaster recovery, you must develop, configure, and test your continuous operation procedures. Continuous

operation includes, but is not limited to, components of the application continuing to function after a subset of their resources is taken offline.

In summary, when a server, or number of servers, fail in your environment you need to ensure that the system continues to function. To do so, the process involves:

- Detecting server failure
- Ensuring that only the functional servers are used to process requests
- Recovering the failed servers
- Configuring the environment to use the recovered servers again

The implementation of this process is different for each component of i2 Analyze, the software prerequisites that are used, and the operating system that the component is deployed on. For some components of i2 Analyze, the process is automatic but you should understand the process and what state the environment is in.

The following information outlines the process for each component in a deployment of i2 Analyze.

Liberty

In a deployment that provides high availability, multiple Liberty servers are used to provide active/active availability. The active/active pattern allows multiple Liberty servers to allow connections from clients and continue to function when some of the Liberty servers fail.

Detecting server failure

To detect if a Liberty server has failed, you can use logs in your load balancer. In the load balancer, monitor the response where you use the `health/live` endpoint to determine which Liberty servers to route clients to. If the response from the `health/live` endpoint for a particular Liberty server is 503, that Liberty server may have failed.

For more information about the endpoint and its responses, see [“The health/live endpoint” on page 275](#).

The Liberty server can return a 503 if it is not started, in the startup process, or there is a temporary loss of connection to another component. Liberty can restart as a non-leader if it temporarily loses the connection to a component and loses leadership. If this is the reason for the 503, the following message is displayed in the `IBM_i2_Component_Availability.log` file:

```
INFO ApplicationStateHandler - I2ANALYZE_STATUS:0066 - Application is entering non-leader mode to serve requests
```

For more information about the log file, see [“Monitor the system availability” on page 278](#).

Automatic failover

If a Liberty server fails, the other Liberty servers continue to function as usual. If a client was connected to the failed Liberty server, the analyst might have to log out from the client and log in again for the load balancer to route the request to one of the live Liberty servers.

If it was the leader Liberty server that failed, one of the remaining Liberty servers is elected leader. For more information about the leadership process, see [“Liberty leadership configuration” on page 274](#).

Recovering failed servers

There are a number of reasons why a server might fail. Use the logs from the failed server to diagnose and solve the issue. For example, you might need to restart the server, increase the hardware specification, or replace hardware components.

- The Liberty logs are in the `deploy\wlp\usr\servers\opal-server\logs` directory.
- The deployment toolkit logs are in the `toolkit\configuration\logs` directory.

For more information about the different log files and their contents, see [Deployment log files](#).

Reinstating high availability

On the recovered Liberty server, run `setup -t startLiberty` to restart the server and i2 Analyze application.

You can use the load balancer logs to ensure that the Liberty server is now returning 200 from the `health/live` endpoint.

Liberty leadership configuration

In a deployment with multiple Liberty servers, one server is determined to be the leader.

Liberty leadership behavior

The leader Liberty can complete actions that require exclusive access to the Information Store database. These actions include;

- Initiating Solr indexing
- Initiating online upgrade tasks
- Initiating alerts for saved Visual Queries
- Initiating deletion-by-rule jobs

Leadership state

ZooKeeper is used to determine and manage the leadership state of the Liberty servers in a deployment. Each Liberty server polls ZooKeeper for the current leadership state. If there is no leader, the Liberty attempts to become the leader. One Liberty will successfully become the leader.

Leadership poll

To become a leader, and to maintain leadership, Liberty must be able to connect to ZooKeeper, Solr, and the database management system. The connection to each component is checked whenever the leadership state is polled. If any of the components cannot be connected to, the Liberty either releases its leadership or cannot run for leader if there is not currently a leader.

The leadership poll schedule is independent on each Liberty server. If the current leader releases its leadership, there is no leader until the leadership poll runs on another Liberty that can then assume leadership status.

The leadership poll is completed on a configurable schedule. By default, the leadership state is checked every five minutes. To change the interval, update the value for the `ServerLeadershipReconnectIntervalMinutes` setting in `DiscoServerSettingsCommon.properties`.

For example:

```
ServerLeadershipReconnectIntervalMinutes=5
```


Error thresholds

The leadership state can change if a certain number of errors are encountered during indexing operations within a specified time span. These types of operations are run only on the leader Liberty. The maximum number of errors that can occur, and the time span in which they can occur, are configurable.

The `MaxPermittedErrorsBeforeLeadershipRelease` setting defines the maximum number of errors that can occur in the time specified by the `TimeSpanForMaxPermittedErrorsBeforeLeadershipReleaseInSeconds` setting. By default, up to five errors can occur within 30 seconds before the Liberty relinquishes its leadership.

For example:

```
MaxPermittedErrorsBeforeLeadershipRelease=5
TimeSpanForMaxPermittedErrorsBeforeLeadershipReleaseInSeconds=30
```

Restarting limits

If the leader Liberty releases leadership, it attempts to restart in non-leader mode. If Liberty cannot connect to each component, it fails to start. By default, it tries to start three times before failing. To start Liberty on this server again, resolve the issue that is stopping Liberty connecting to all of the components and run `setup -t startLiberty`.

You can change the maximum number of times that Liberty attempts to start before it fails. To change the number of attempts, update the value of the `ServerLeadershipMaxFailedStarts` setting in `DiscoServerSettingsCommon.properties`.

```
ServerLeadershipMaxFailedStarts=3
```

The health/live endpoint

The `api/v1/health/live` REST endpoint returns the status of the i2 Analyze application running on a Liberty server.

The `api/v1/health/live` endpoint can return one of two possible responses:

- 200 The application is UP and handling requests.
- 503 The application is DOWN and not handling requests.

When the endpoint is called, the application checks whether it can connect to ZooKeeper, Solr, and the database management system if your deployment uses one. If it can connect to all of the components, the endpoint returns 200. This means that you can route client requests to this Liberty server.

If the application fails to connect to a component, the endpoint returns 503. The Liberty server cannot process client requests.

The endpoint also returns 503 until the application completes the start process.

When you configure a load balancer for use in a high availability deployment, it should use the `api/v1/health/live` endpoint to determine which servers to send requests to. For more information, see [Deploying a load balancer](#).

Database management system

In a deployment that provides high availability, the Information Store database is deployed in an active/passive pattern. In the active/passive pattern, there is a primary database instance and a number of standbys.

Detecting server failure

There are a number of different ways to detect if there has been a database failure.

- The `IBM_i2_Component_Availability.log` contains messages that report the connection status for the Information Store database. For more information about the messages that are specific to the database, see [“Database management system messages” on page 281](#).

There are also a number of different tools that are specific to your database management system.

- Db2:
 - Db2 fault monitor on Linux
 - Heartbeat monitoring in clustered environments
 - Monitoring Db2 High Availability Disaster Recovery (HADR) databases

For more information about detecting failures for Db2, see [Detecting an unplanned outage](#).

- SQL Server
 - To understand when SQL Server initiates failover, see [How Automatic Failover Works](#).

Manual and automatic failover

When the primary server fails, the system must fail over and use the remaining servers to complete operations.

- Db2
 - If you are using an automated-cluster controller, failover to a standby instance is automatic. For more information, see .
 - If you are using client-side automatic client rerouting, then you must manually force a standby instance to become the new primary. For more information about initiating a takeover, see [Performing an HADR failover operation](#).
- SQL Server
 - When SQL Server is configured for high availability in an availability group with three servers, failover is automatic. For more information about failover, see [Automatic Failover](#).

Recovering failed servers

There are a number of reasons why a server might fail. You can use the logs from the database server to diagnose and solve the issue. For example, you might need to restart the server, increase the hardware specification, or replace hardware components that caused the issue.

When the server is back online and functional, you can recover it to become the primary server again. Alternatively, you can recover it to become the standby for the new primary that you failed over to.

This might include recovering a back up of the Information Store database from before the server failed.

Reinstating high availability

- Db2

- Some toolkit tasks only work on the original primary database server when you are not using an automated-cluster controller such as TSAMP. To use those toolkit tasks, you must revert to using the original primary database server after a failure or redeploy your system to use the new primary database server.
 - For more information about the process to make the recovered database the primary again, see [Reintegrating a database after a takeover operation](#).
 - To redeploy with the new primary, update the `topology.xml` on each Liberty to reference the new server in the `host-name` and `port-number` and redeploy each Liberty.
- SQL Server
 - On SQL Server it is not required to return to the previous primary, however you might choose to do so. You can initiate a planned manual failover to return the initial primary server. For more information, see [Planned Manual Failover \(Without Data Loss\)](#).

Solr

In a deployment that provides high availability, i2 Analyze uses SolrCloud to deploy Solr with fault tolerance and high availability capabilities. Solr uses an active/active pattern that allows all Solr servers to respond to requests.

Detecting server failure

To detect if there has been a Solr failure, you can use the following mechanisms:

- The `IBM_i2_Component_Availability.log` contains messages that report the status of the Solr cluster. For more information about the messages that are specific to Solr, see [“Solr messages” on page 279](#).
- The Solr Web UI displays the status of any collections in a Solr cluster. Navigate to the following URL in a web browser to access the UI: `http://localhost:8983/solr/#/~cloud?view=graph`. Where `localhost` and `8983` are replaced with the hostname and port number of one of your Solr nodes.

Automatic failover

Solr continues to update the index and provide search results while there is at least one replica available for each shard.

Recovering failed servers

There are a number of reasons why a server might fail. Use the logs from the server to diagnose and solve the issue.

Solr logs:

- The Solr logs are located in the `i2analyze\deploy\solr\server\logs\8983` directory on each Solr server. Where `8983` is the port number of the Solr node on the server.
- The application logs are in the `deploy\wlp\usr\servers\opal-server\logs` directory, specifically the `IBM_i2_Solr.log` file.

For more information about the different log files and their contents, see [Deployment log files](#).

To recover the failed server you might need to restart the server, increase the hardware specification, or replace hardware components.

Reinstating high availability

On the recovered Solr server, run `setup -t startSolrNodes -hn solr.hostname` to start the Solr nodes on the server.

You can use the `IBM_i2_Component_Availability.log` and Solr Web UI to ensure that the nodes start correctly and that the cluster returns to its previous state.

ZooKeeper

In a deployment that provides high availability, ZooKeeper is used to manage the Solr cluster and maintain Liberty leadership information. ZooKeeper uses an active/active pattern that allows all ZooKeeper servers to respond to requests.

Detecting server failure

To detect ZooKeeper server failures:

- The `IBM_i2_Component_Availability.log` contains messages that report the status of the ZooKeeper quorum. For more information, see [“ZooKeeper status messages”](#) on page 280.
- The Solr Web UI contains the status of the ZooKeeper quorum: `http://localhost:8983/solr/#/~cloud?view=zkstatus`. Where *localhost* and *8983* are replaced with the host name and port number of one of your Solr nodes.

Manual and automatic fail over

The ZooKeeper quorum continues to function while more than half of the total number of ZooKeeper hosts in the quorum are available. For example, if you have three ZooKeeper servers, your system can sustain one ZooKeeper server failure.

Recovering failed servers

There are a number of reasons why a server might fail. Use the logs from the server to diagnose and solve the issue.

ZooKeeper logs:

- The ZooKeeper logs are located in the `i2analyze\data\zookeeper\8\ensembles\zoo\zkhosts\1\logs` directory on the failed ZooKeeper server. Where *1* is the identifier of the ZooKeeper host on the server.
- The application logs are in the `deploy\wlp\usr\servers\opal-server\logs` directory.

For more information about the different log files and their contents, see [Deployment log files](#).

To recover the failed server you might need to restart the server, increase the hardware specification, or replace hardware components.

Reinstating high availability

On the recovered ZooKeeper server, run `setup -t startZkHosts -hn zookeeper.hostname` to start the ZooKeeper host on the server.

You can use the `IBM_i2_Component_Availability.log` and Solr Web UI ZooKeeper status view to ensure that the system returns to its previous state.

Monitor the system availability

i2 Analyze reports on the availability of each component in the environment so that the state can be monitored. The availability is logged on each Liberty server in the environment.

To monitor the status of your deployment, use the `IBM_i2_Component_Availability.log` file. This log file is located in the `wlp/servers/opal-server/opal-services/logs` directory on each Liberty server in your deployment. Therefore, you must monitor the log file on each Liberty server in your deployment.

The log contains messages about the Liberty leadership status, Solr cluster status, and the availability of each component.

Liberty leadership messages

If the Liberty starting the election process, the following message is displayed:

- INFO ServerLeadershipMonitor - We are running for Liberty leader

If the Liberty is elected leader, the following messages are displayed:

- INFO ServerLeadershipMonitor - We are the Liberty leader
- INFO ApplicationStateHandler - I2ANALYZE_STATUS:0065 - Application is entering leader mode

If the Liberty is not elected leader, the following messages are displayed:

- INFO ServerLeadershipMonitor - We are not the Liberty leader
- INFO ApplicationStateHandler - I2ANALYZE_STATUS:0063 - Application is entering non-leader mode to serve requests

Component messages

If all of the components are available to the Liberty server, the following message is displayed:

- INFO ComponentAvailabilityCheck - All components are available

If at least one of the components is not available to the Liberty server, the following message is displayed:

- WARN ComponentAvailabilityCheck - Not all components are available

When the Liberty server continues to check the availability of each component, the following message is displayed:

- INFO ComponentAvailabilityCheck - I2ANALYZE_STATUS:0068 - The application is waiting for all components to be available

Solr messages

If the Liberty server cannot connect to the Solr cluster, the following message is displayed:

- WARN ComponentAvailabilityCheck - Unable to connect to Solr cluster
- WARN ComponentAvailabilityCheck - The Solr cluster is unavailable

The following messages describe the state of the Solr cluster in the deployment:

- ALL_REPLICAS_ACTIVE - The named Solr collection is healthy. All replicas in the collection are active.

For example:

- INFO SolrHealthStatusLogger - 'main_index','ALL_REPLICAS_ACTIVE'

- **DEGRADED** - The named Solr collection is degraded. The minimum replication factor can be achieved, but at least one replica is down or failed to recover.

For example:

```
WARN SolrHealthStatusLogger      - 'main_index','DEGRADED'
```

When the status is DEGRADED, data can still be written to the Solr index. When the status returns to ALL_REPLICAS_ACTIVE, the data is synchronized in the Solr index as described in [“Data synchronization in i2 Analyze” on page 281](#).

The deployment can still be used with the collection in a degraded state, however the deployment can now sustain fewer Solr server failures. If a degraded state is common, or lasts for an extended time, you should investigate the Solr logs to improve the stability of the system.

- **RECOVERING** - The named Solr collection is recovering. The minimum replication factor cannot be achieved, because too many replicas are currently in recovery mode.

For example:

```
INFO SolrHealthStatusLogger      - 'main_index','RECOVERING'
```

If all of the replicas recover, the status changes to ALL_REPLICAS_ACTIVE. If the replicas fail to recover, the status changes to DEGRADED or DOWN.

- **DOWN** - The named Solr collection is down. The minimum replication factor cannot be achieved because too many replicas are down or have failed to recover.

For example:

```
WARN SolrHealthStatusLogger      - 'main_index','DOWN'
```

When the status is DOWN, data cannot be written to the Solr index. You should attempt to resolve the issue. For more information, see [“Solr” on page 277](#).

- **UNAVAILABLE** - The named Solr collection is unavailable. The application cannot connect to the collection.

For example:

```
WARN SolrHealthStatusLogger      - 'main_index','UNAVAILABLE'
```

When the status is UNAVAILABLE, data cannot be written to the Solr index. You should attempt to resolve the issue. For more information, see [“Solr” on page 277](#).

ZooKeeper status messages

If the connection to ZooKeeper is lost, the following messages are displayed:

```
WARN ComponentAvailabilityCheck  - Unable to connect to ZooKeeper
WARN ComponentAvailabilityCheck  - The ZooKeeper quorum is unavailable
```

When the connection to the database is restored, the all components are active message is displayed.

Database management system messages

If the connection to the database is lost, the following messages are displayed:

```
WARN ComponentAvailabilityCheck - Information Store database appears to be offline,
retrying...
WARN ComponentAvailabilityCheck - Unable to connect to the Information Store database.
java.sql.SQLException: The TCP/IP connection to the host has failed.
Error: "Socket operation on nonsocket: configureBlocking. Verify the connection properties.
Make sure that an instance of SQL Server is running on the host and accepting TCP/IP
connections at the port.
Make sure that TCP connections to the port are not blocked by a firewall.". DSRA0010E: SQL
State = 08S01, Error Code = 0
WARN ComponentAvailabilityCheck - The Information Store database is unavailable
```

When the connection to the database is restored, the all components are active message is displayed.

Data synchronization in i2 Analyze

In a deployment of i2 Analyze that contains an Information Store database, data is stored in both the database and the Solr search index. In i2 Analyze, the database is considered the source of truth for data in the deployment. When data is added, modified, or removed from the deployment, the data change is applied to the database before those changes are reflected in the search index.

When data changes occur, the database and Solr retain watermarks that record if the operation was successfully completed in the database and Solr index. i2 Analyze routinely checks these watermarks in the deployment. i2 Analyze can respond to the state of the deployment in a number of different ways, and can normally be resolved without manual intervention.

Note: There is no mechanism in i2 Analyze to ensure that data is written to any standby database instances before the watermarks are updated.

Changes to the Solr collection health

If at least on replica for a shard is unavailable, the collection is marked as unhealthy. If there are data changes in the database during this time, the Solr index is updated on the replicas that are available while the minimum replication factor is achieved.

When all of the replicas are available and the Solr collection is marked as healthy again, all of the data changes that occurred when it was in the unhealthy state are reindexed. This means that all data changes are indexed on every replica.

At i2 Analyze application start

When the i2 Analyze application starts, i2 Analyze checks the watermarks.

- If the deployment is in sync, then the deployment starts as usual.
- If there is data in the database that isn't in the Solr index, the indexing process indexes the data from the database that is not in the Solr index.
- If there is data in the Solr index that isn't in the database, i2 Analyze does not start and the following message is displayed:

```
The 'main_index' index reports that it is ahead of the database
```

This situation can occur if you recover the database from a backup that was taken before the current Solr index. In your back up and restore procedure, ensure that you restore the database from a backup that was taken after the Solr index backup.

Implementation details

The values that i2 Analyze uses to record the state of the deployment are called *watermarks*. Whenever data is changed in the database, the data change is then indexed in Solr. When Solr has completed indexing the changed data, the watermarks are updated. The watermarks are stored in the database and ZooKeeper. Two watermarks are maintained, a *high watermark* and a *low watermark*.

High watermark

The high watermark is updated in both locations when Solr achieves the minimum replication factor. That is, when the index is updated on a specified number of replicas for a shard. When you configure Solr for high availability, you specify the minimum replication factor for each Solr collection.

Low watermark

The low watermark is updated in both locations when Solr achieves the maximum replication factor. That is, when the index is updated on all of the replicas for a shard. The low watermark is only updated after the high watermark, the low watermark cannot have a higher value than the high watermark.

Configuration resources

The configuration resources section contains information that is referenced elsewhere in the configuration section. The section also includes the reference documentation for the configuration files that are in the i2 Analyze deployment toolkit.

Configuration files reference

In the deployment toolkit, you can identify and modify the core components of your deployment. Many of the settings in the configuration files must be set before you can deploy i2 Analyze.

The credentials.properties file

To allow the deployment toolkit to update the database, Lightweight Third-Party Authentication (LTPA) keys, and Solr search platform components, you must provide user names and passwords. The user names and passwords that you provide allow the system to set up and administer components of i2 Analyze, and are not used to access i2 Analyze.

Three types of credentials are stored in `credentials.properties`:

- Database
- LTPA keys
- Solr search platform

After you specify the credentials, you might need to change the values in the `credentials.properties` file from the values that are used in development or test deployments. For example, your database management system might require a different user name and password in a test or production system.

When you deploy i2 Analyze, the passwords in the `credentials.properties` file are encoded.

Database

For each database that is identified in `topology.xml`, you must specify a user name and a password in the `credentials.properties` file. The setup script uses this information to authenticate with the database.

Note: The user that you specify must have privileges to create and populate databases in the database management system.

The database credentials are stored in the following format:

```
db.infostore.user-name=user name
db.infostore.password=password
```

For example:

```
db.infostore.user-name=admin
db.infostore.password=password
```

Note: The `db.truststore.password` credential is used only when you configure the connection between the database and Liberty to use SSL. If you are not using SSL to secure this connection, you do not need to specify a value for the `db.truststore.password` credential. For more information about configuring SSL, see [Configure Secure Sockets Layer with i2 Analyze](#).

LTPA keys

You must provide a value for the `ltpakeys.password` property. This value is used by the system to encrypt LTPA tokens.

- For a stand-alone deployment of i2 Analyze, you can specify any value as the password.
- For a deployment of i2 Analyze that uses LTPA tokens to authenticate with other systems, you must specify the same password that those systems use.

Solr search platform

The Solr search platform is used to search data in the Information Store. You must provide values for the `solr.user-name` and `solr.password` properties. Any Solr indexes are created when i2 Analyze is first deployed, and the values that you provide here become the Solr user name and password.

If you have already deployed i2 Analyze, and you want to change the Solr password, new properties must be created in the following format:

```
solr.user-name.new=value
solr.password.new=value
```

For example:

```
#Solr credentials
#The user name and password for solr to use once deployed
solr.user-name=admin
solr.password={enc}E3FGHjYUI2A\=

solr.user-name.new=admin1
solr.password.new=password
```

As a part of the deployment process:

- The new values replace the original values in the file
- The password value is encoded
- Once changed, the `credentials.properties` file is cleaned, removing the duplicate values

After you update the credentials, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes.

The environment.properties file

The script in the deployment toolkit that deploys i2 Analyze requires information about the deployment environment. In the environment properties you can identify your database management system location, specify the installation directories that i2 Analyze uses, and specify the locations where i2 Analyze can store data.

The following properties are in the `environment.properties` file:

db.installation.dir

Database management system The installation path of the database management system or the database management system client. If you are using IBM Db2, for example `C:/Program Files/IBM/SQLLIB` on Windows, or `/opt/ibm/db2/V11.5` on Linux If you are using Microsoft SQL Server, for example `C:/Program Files/Microsoft SQL Server/Client SDK/ODBC/170` on Windows, or `/opt/mssql-tools` on Linux

db.database.location.dir

The root path of the database file storage. If you are using IBM Db2, for example `C:` on Windows, or `/home/db2inst1` on Linux If you are using Microsoft SQL Server, for example `C:/Program Files/Microsoft SQL Server/MSSQL15.MSSQLSERVER/MSSQL/DATA` on Windows, or `/var/opt/mssql/data` on Linux

wlp.home.dir

Application server The installation path for WebSphere Liberty Profile.

For example, `C:/IBM/i2analyze/deploy/wlp` on Windows, or `/opt/IBM/i2analyze/deploy/wlp` on Linux

By default, the value is `/opt/IBM/i2analyze/deploy/wlp`

solr.home.dir

The installation path for Apache Solr. (Required for the Opal deployment pattern)

For example, `C:/IBM/i2analyze/deploy/solr` on Windows or `/opt/IBM/i2analyze/deploy/solr` on Linux

By default, the value is `/opt/IBM/i2analyze/deploy/solr`

zookeeper.home.dir

The installation path for Apache ZooKeeper. (Required for the Opal deployment pattern)

For example, `C:/IBM/i2analyze/deploy/zookeeper` on Windows or `/opt/IBM/i2analyze/deploy/zookeeper` on Linux

By default, the value is `/opt/IBM/i2analyze/deploy/zookeeper`

apollo.data

The path to a directory where i2 Analyze can store files. By default, the directory is also used to store the Solr index files.

For example, C:/IBM/i2analyze/data on Windows, or /opt/IBM/i2analyze/data on Linux

By default, the value is /opt/IBM/i2analyze/data

backup.config.location.dir

Backup and restore The location where the i2 Analyze configuration backups are created and restored from. This location must exist on the Liberty server where you run the backupConfiguration command.

backup.solr.location.dir

The location where the Solr index and ZooKeeper configuration backups are created and restored from. This location must be accessible by every Solr node in your deployment.

backup.db.location.dir

The location where the database backup are created and restored from. The user that is specified in the credentials.properties file for your database management system must have write permissions to this location.

The topology.xml file

In the deployment toolkit, the topology file defines the physical architecture of the finished deployment, and the components that make up the pieces of that architecture. The topology file contains domain-specific information that you must provide before deployment can take place.

The topology.xml file contains the information that the setup script requires to identify the prerequisites that are being used, where the components of the system are located, and the fragments to combine to create the application.

The topology.xml file contains distinct sections for defining different aspects of a deployment.

i2-data-sources

A definition of each data source that is available within the deployment. For more information, see [“Data sources” on page 286](#).

databases

The names, types, and locations of the databases that i2 Analyze uses to store data. For more information, see [“Databases” on page 286](#).

applications

The applications that comprise this deployment of i2 Analyze, and the locations of the application servers on which they are to be installed. For more information, see [“Applications” on page 288](#).

zookeepers and solr-clusters

The ZooKeeper hosts and Solr clusters that are used in this deployment. For more information, see [“Solr and ZooKeeper” on page 291](#).

connectors

The connectors that are used with the i2 Connect gateway in this deployment. For more information, see [“Connectors” on page 295](#).

After you update the topology file, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes. For more information about redeploying your system, see [“Redeploying Liberty” on page 304](#).

Data sources

The topology files that are supplied with each example in the deployment toolkit contains a preconfigured `<i2-data-source>` element.

The `<i2-data-sources>` element contains a single `<i2-data-source>` element for the data source that the deployment connects to. For example:

```
<i2-data-sources>
  <i2-data-source default="false" id="infostore">
    <DataSource Version="0" Id="">
      <Name>Information Store</Name>
    </DataSource>
  </i2-data-source>
</i2-data-sources>
```

Where:

Attribute	Description
id	A unique identifier that is used to distinguish this data source throughout the system.
default	An optional attribute. The value must be <code>false</code> in this version of i2 Analyze.

Each `<i2-data-source>` contains a single `<DataSource>` element that has two standard attributes and two child elements.

Attribute	Description
Id	Reserved for future use. The value must be empty in this version of i2 Analyze.
Version	Reserved for future use. The value must be <code>0</code> in this version of i2 Analyze.

Element	Description
Name	The name of this data source, which is presented to users.

Databases

The `<databases>` element defines the database that i2 Analyze uses to store data. The `<database>` element contains attributes that define information about the database, and the mechanism that is used to connect to it.

For example:

```
<database database-type="InfoStore"
  dialect="db2" database-name="ISTORE" instance-name="DB2"
  xa="false" host-name="host" id="infostore"
  port-number="50000" />
```

Where:

Attribute	Description
database-type	The value must be InfoStore in this version of i2 Analyze.
dialect	Specifies the type of database engine. This attribute can be set to one of the following values: <ul style="list-style-type: none"> • db2 • sqlserver
database-name	A name that identifies the database to the database engine.
instance-name	The instance name that was specified during installation of your database engine. If you are using SQL Server, you can either specify the port-number or instance-name attribute. For SQL Server on Linux, you must use the port-number attribute.
xa	Determines whether distributed transactions are enabled for this database.
host-name	The hostname of the server where the database is located.
id	A unique identifier that is used to distinguish this database throughout the system. The value must match the id in the <i2-data-source> element.
port-number	The port on the database server to send requests to. If you are using SQL Server, you can either specify the port-number or instance-name attribute.

Remote database attributes

If your database is on a remote server from the i2 Analyze application, you must use the following attributes in the topology.xml file:

Attribute	Description
os-type	Identifies the operating system of the server where the database is located. This can be one of the following values: <ul style="list-style-type: none"> • WIN • UNIX If you are using Db2, you can also specify AIX.
node-name	Identifies the node to create in the Db2 node directory. The value of the node-name attribute must start with a letter, and have fewer than 8 characters. For more information about naming in Db2, see Naming conventions .

For example, if you are using IBM Db2:

```
<database database-type="InfoStore" dialect="db2" instance-name="DB2"
  database-name="ISTORE" xa="false" id="infostore"
  host-name="remote.hostname" port-number="50000"
  node-name="node1" os-type="WIN" />
```

For example, if you are using Microsoft SQL Server:

```
<database database-type="InfoStore" dialect="sqlserver" instance-name="SQLSERVER"
  database-name="ISTORE" xa="false" id="infostore"
  host-name="remote.hostname" os-type="WIN" />
```

Applications

The <applications> and child <application> element within the topology file define the indexes, file stores, and WAR file for the application.

An <application> element has the following attributes:

Attribute	Description
name	The value must be opal-server in this version of i2 Analyze.
host-name	The hostname of the server where the application is located.
http-server-host	Determines whether the HTTP Server is configured by the deployment toolkit. Set to <i>true</i> to configure the HTTP Server, or <i>false</i> not to configure the HTTP Server.

The following sections explain how to define the indexes, file stores, and WAR file for an application.

File stores

The <application> element contains a child <file-stores> element that defines the file stores that are used by the application. The *location* attribute specifies the file path to use for each file store. The other attributes must be left with their default values.

WAR files

The <application> element contains a child <wars> element. The <wars> element contains child <war> elements that define the contents of the i2 Analyze WAR files that are installed on the application server.

Each <war> element has the following attributes:

Attribute	Description
target	The type of WAR file to create. The following types are available: <ul style="list-style-type: none">opal-services-isopal-services-daodopal-services-is-daod
name	The name of the WAR file.

Attribute	Description
i2-data-source-id	A name that identifies the <i2-data-source> element for this WAR file.
context-root	The URL that is used to access the WAR file. <ul style="list-style-type: none"> • For the opal-services-is and opal-services-is-daod WARs, this is opal by default. • For the opal-services-daod WAR, this is opaldaod by default.

The following child elements of the <war> element can be defined:

data-sources

Identifies the database that is used by this WAR in a child <data-source> element. The <data-source> element has the following attributes:

Attribute	Description
database-id	Identifies the database to use. This value must match the id value that is specified in a <database> element. For the opal-services-is and opal-services-is-daod WARs, the <data-source> element must reference a database of type InfoStore.
create-database	Determines whether the database is to be created on the server that is specified in the <database> element. Set as <i>true</i> to create the database, or <i>false</i> not to create the database.

solr-collection-ids

Identifies the Solr collections that are used by this WAR in child <solr-collection-id> elements. Each Solr collection is identified by the collection-id attribute, the value of which must match the id value that is specified in a <solr-collection> element. For more information about the collections that you must reference, see [<solr-collections>](#).

A Solr collection belongs to a Solr cluster. Each Solr cluster is identified by the cluster-id attribute, the value of which must match an id value that is specified in a <solr-cluster> element.

For more information about the ZooKeeper and Solr-related elements in the topology.xml file, see [“Solr and ZooKeeper”](#) on page 291.

file-store-ids

Identifies the file stores that are used by this WAR. Each file store is identified by the value attribute, the value of which must match an id value that is specified in a <file-store> element.

connector-ids

Identifies the connectors that are available in the opal-services-is-daod and opal-services-daod WARs in a child <connector-id> element. Each connector is identified by the value attribute, the value of which must match an id value that is specified in a <connector> element.

For more information about connector-related elements in the topology.xml file, see [“Connectors”](#) on page 295.

fragments

Specifies the fragments that are combined to create the WAR.

Note: All WARs must contain the common fragment.

In the example topology.xml file for the information-store-opal example deployment, the opal-server application definition is:

```
<application name="opal-server" host-name="" http-server-host="true">
  <wars>
    <war context-root="opal" name="opal-services-is" i2-data-source-id="infostore"
      target="opal-services-is">
      <data-sources>
        <data-source database-id="infostore" create-database="true" />
      </data-sources>
      <file-store-ids>
        <file-store-id value="chart-store" />
        <file-store-id value="job-store" />
        <file-store-id value="recordgroup-store" />
      </file-store-ids>
      <fragments>
        <fragment name="opal-services-is" />
        <fragment name="opal-services" />
        <fragment name="common" />
        <fragment name="default-user-profile-provider" />
      </fragments>
      <solr-collection-ids>
        <solr-collection-id collection-id="main_index" cluster-id="is_cluster" />
        <solr-collection-id collection-id="match_index1" cluster-id="is_cluster" />
        <solr-collection-id collection-id="match_index2" cluster-id="is_cluster" />
        <solr-collection-id collection-id="highlight_index" cluster-id="is_cluster" />
        <solr-collection-id collection-id="chart_index" cluster-id="is_cluster" />
      </solr-collection-ids>
    </war>
  </wars>
  <file-stores>
    <file-store location="" id="job-store" type="job-store" />
    <file-store location="" id="recordgroup-store" type="recordgroup-store" />
  </file-stores>
</application>
```


In the example `topology.xml` file for the `daod-opal` deployment, the `opal-server` application definition is:

```
<application http-server-host="false" name="opal-server" host-name="">
  <wars>
    <war context-root="opaldaod" name="opal-services-daod" i2-data-source-id="opalDAOD"
target="opal-services-daod">
      <file-store-ids>
        <file-store-id value="recordgroup-daod-store" />
      </file-store-ids>
      <fragments>
        <fragment name="opal-services" />
        <fragment name="common" />
      </fragments>
      <solr-collection-ids>
        <solr-collection-id collection-id="daod_index" cluster-id="is_cluster"/>
      </solr-collection-ids>
    </war>
  </wars>
  <file-stores>
    <file-store location="" id="recordgroup-daod-store" type="recordgroup-store" />
  </file-stores>
</application>
```

Solr and ZooKeeper

i2 Analyze uses Solr for text indexing and search capabilities. The ZooKeeper service maintains configuration information and distributed synchronization across Solr and Liberty.

The `topology.xml` file for a deployment that includes the `opal-server` application also includes the `<zookeepers>` and `<solr-clusters>` elements. The `<solr-clusters>` and `<zookeepers>` elements define the Solr cluster that is used in a deployment and the ZooKeeper instance that manages it.

<solr-clusters>

In the supplied topology.xml file that includes the opa1-server application with the opa1-services-is WAR, the <solr-clusters> definition is:

```
<solr-clusters>
  <solr-cluster id="is_cluster" zookeeper-id="zoo">
    <solr-collections>
      <solr-collection
        id="main_index" type="main"
        lucene-match-version=""
        max-shards-per-node="4" num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="match_index1" type="match"
        lucene-match-version=""
        max-shards-per-node="4" num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="match_index2" type="match"
        lucene-match-version=""
        max-shards-per-node="4" num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="highlight_index" type="highlight"
        lucene-match-version=""
        max-shards-per-node="4" num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="chart_index" type="chart"
        lucene-match-version=""
        max-shards-per-node="4" num-shards="4" num-replicas="1"
      />
    </solr-collections>
    <solr-nodes>
      <solr-node
        memory="2g"
        id="node1"
        host-name=""
        data-dir=""
        port-number="8983"
      />
    </solr-nodes>
  </solr-cluster>
</solr-clusters>
```

The <solr-clusters> element includes a child <solr-cluster> element. The id attribute of the <solr-cluster> element is a unique identifier for the Solr cluster. To associate the Solr cluster with the ZooKeeper instance, the value of the zookeeper-id attribute must match the value of the id attribute of the <zookeeper> element.

<solr-collections>

The <solr-collections> element is a child of the <solr-cluster> element. The <solr-collections> element has child <solr-collection> elements.

Depending on the WARs that are included in the application, the number and type of required child <solr-collection> elements is different.

opal-services-is

In the opal-services-is WAR, you must have <solr-collection> elements with each of the following values for the type attribute:

- main - you must have either one or two collections of type main
- match - you must have two collections of type match
- highlight - you must have one collection of type highlight
- chart - you must have one collection of type chart

opal-services-daod

In the opal-services-daod WAR, you must have one <solr-collection> element with a value of daod for the type attribute.

opal-services-is-daod

In the opal-services-is-daod WAR, you must have <solr-collection> elements with each of the following values for the type attribute:

- main - you must have either one or two collections of type main
- daod - you must have one collection of type daod
- match - you must have two collections of type match
- highlight - you must have one collection of type highlight
- chart - you must have one collection of type chart

The <solr-collection> element has the following attributes:

Attribute	Description
id	An identifier that is used to identify the Solr collection.
type	The type of the collection. The possible values are: <ul style="list-style-type: none">• main• daod• match• highlight• chart
lucene-match-version	The Lucene version that is used for the collection. At this release, the value is populated when you deploy i2 Analyze.
num-shards	The number of logical shards that are created as part of the Solr collection.
num-replicas	The number of physical replicas that are created for each logical shard in the Solr collection.

Attribute	Description
max-shards-per-node	The maximum number of shards that are allowed on each Solr node. This value is the result of <i>num-shards</i> multiplied by <i>num-replicas</i> .
min-replication-factor	The minimum number of replicas that an update must be replicated to for the operation to succeed. This value must be greater than 0 and less than or equal to the value of <i>num-replicas</i> . This attribute is optional.
num-csv-write-threads	The number of threads that are used to read from the database and write to the temporary .csv file when indexing data in the Information Store. This attribute is optional, and applies to Solr collections of type <i>main</i> and <i>match</i> only. The total of <i>num-csv-write-threads</i> and <i>num-csv-read-threads</i> must be less than the number of cores available on the Liberty server.
num-csv-read-threads	The number of threads that are used to read from the temporary .csv file and write to the index when indexing data in the Information Store. This attribute is optional, and applies to Solr collections of type <i>main</i> and <i>match</i> only. This value must be less than the value of <i>num-shards</i> . The total of <i>num-csv-write-threads</i> and <i>num-csv-read-threads</i> must be less than the number of cores available on the Liberty server.

<solr-nodes>

The <solr-nodes> element is a child of the <solr-cluster> element. The <solr-nodes> element can have one or more child <solr-node> elements. Each <solr-node> element has the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the Solr node.
memory	The amount of memory that can be used by the Solr node.
host-name	The hostname of the Solr node.
data-dir	The location that Solr stores the index.
port-number	The port number of the Solr node.

<zookeepers>

In the supplied `topology.xml` file that includes the `opal-server` application, the `<zookeepers>` definition is:

```
<zookeepers>
  <zookeeper id="zoo">
    <zkhhosts>
      <zkhost
        id="1"
        host-name=""
        data-dir=""
        port-number="9983"
        quorum-port-number=""
        leader-port-number=""
      />
    </zkhhosts>
  </zookeeper>
</zookeepers>
```

The `<zookeepers>` element includes a child `<zookeeper>` element. The `id` attribute of the `<zookeeper>` element is a unique identifier for the ZooKeeper instance. To associate the ZooKeeper instance with the Solr cluster, the value of the `id` attribute must match the value of the `zookeeper-id` attribute of the `<solr-cluster>` element.

The `<zkhhosts>` element is a child of the `<zookeeper>` element. The `<zkhhosts>` element can have one or more child `<zkhost>` elements. Each `<zkhost>` element has the following attributes:

Attribute	Description
<code>id</code>	A unique identifier that is used to identify the ZooKeeper host. This value must be an integer in the range 1 - 255.
<code>host-name</code>	The hostname of the ZooKeeper host.
<code>data-dir</code>	The location that ZooKeeper uses to store data.
<code>port-number</code>	The port number of the ZooKeeper host.
<code>quorum-port-number</code>	The port number that is used for ZooKeeper quorum communication. By default, the value is 10483.
<code>leader-port-number</code>	This port number that is used by ZooKeeper for leader election communication. By default, the value is 10983.

Connectors

An `i2 Analyze` deployment that includes the `i2 Connect` gateway enables you to connect to external data sources.

The `topology.xml` file for a deployment that includes the `opal-server` application with the `opal-services-is-daod` or `opal-services-daod` WARs includes the `<connectors>` element. The `<connectors>` element defines the connectors that are used in a deployment.

<connectors>

The <connectors> element contains one or more child <connector> elements. The <connector> element can have the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the connector.
name	The name of the connector, which is presented to users.
base-url	The URL to the connector, made up of host name and port number. For example, <code>https://host name:port number</code> . You can use the HTTP or HTTPS protocol.
configuration-url	The URL to any configuration that is required for the connector. The default value for the <code>configuration-url</code> attribute is <code>/config</code> . The presence of the attribute in the <code>topology.xml</code> file is optional. If it is not present, the default value is used.
gateway-schema	The short name of the gateway schema whose item types the connector can use, overriding any setting in the connector configuration. This attribute is optional; when present its value must match one of the settings in <code>ApolloServerSettingsMandatory.properties</code> .
schema-short-name	The short name by which the connector schema for the connector is known in the rest of the deployment, overriding any setting in the connector configuration. Use this optional attribute to avoid naming collisions, or to change the name that users see in client software.

After a successful deployment, the `topology.xml` file in the `daod-opa1`, `chart-storage-daod`, and `information-store-daod-opa1` example configurations contains the following <connectors> definition:

```
<connectors>
  <connector id="example-connector" name="Example"
    base-url="http://localhost:3700/" />
</connector>
```

The http-server.properties file

The values that describe your HTTP server installation are stored in the `http-server.properties` file. You can modify the values to match the deployment environment.

The following settings are in the `http-server.properties` file:

http.server.home.dir

The path to the IBM HTTP Server installation directory.

For example, `C:/IBM/HTTPServer` on Windows, or `/opt/IBM/HTTPServer` on Linux

http.was.module.dir

The path to the WebSphere Plugin for IBM HTTP Server.

For example, C:/IBM/HTTPServer/plugin/bin on Windows, or /opt/IBM/HTTPServer/plugin/bin on Linux.

http.server.ssl.enabled

The status of SSL mode on the HTTP server. When this setting is true, you must also set `http.server.keystore.file`

By default, the value is `false`

http.server.ssl.require.secure.backend

Controls whether the WebSphere Plugin for IBM HTTP Server requires secured connections to backend transports, for example WebSphere Liberty Profile. When set to `false`, the plugin can use an insecure transport if there is no secure connection available. Change this to `true` to ensure that the WebSphere plugin uses a secure connection to backend transports. When set to `true`, a secure connection to the backend must be available.

By default, the value is `false`

http.server.keystore.file

The path to the key database for the HTTP server. This setting is only required if SSL is enabled.

http.server.keystore.certificate.label

The label of the certificate in the key database for the HTTP Server. This setting is only required if SSL is enabled.

http.server.log.level

The verbosity of the messages recorded in the IBM HTTP Server error logs. When a particular level is specified, messages from all other levels of higher significance are reported as well. For example, when `INFO` is specified, messages with log levels of `NOTICE` and `WARN` are also recorded.

By default, the value is `ERROR`

http.server.http.port.number

The port number that is used to connect to the IBM HTTP Server if SSL is disabled.

By default, the value is `80`.

http.server.https.port.number

The port number that is used to connect to the IBM HTTP Server if SSL is enabled.

By default, the value is `443`.

The ApolloServerSettingsMandatory.properties file

The following properties are in the `ApolloServerSettingsMandatory.properties` file:

SchemaResource

The file that contains the schema for the Chart Store or the Information Store. This setting can be removed if the deployment includes neither store. When specified, this XML file needs to be on the classpath.

By default, the value is `law-enforcement-schema.xml`

ChartingSchemesResource

The file that contains charting schemes for the Chart Store or the Information Store. This setting can be removed if the deployment includes neither store. When specified, this XML file needs to be on the classpath.

By default, the value is law-enforcement-schema-charting-schemes.xml

Gateway.External.SchemaResource

A file that contains a gateway schema for this deployment of i2 Analyze. This setting can be removed if the deployment does not require a gateway schema. When specified, this XML file needs to be on the classpath.

Gateway.External.ChartingSchemesResource

A file that contains gateway charting schemes for this deployment of i2 Analyze. This setting can be removed if the deployment does not include a gateway schema. When specified, this XML file needs to be on the classpath.

UserGroupsProvider

The full class name of the user group provider. To retrieve user groups from the WebSphere user registry, specify `com.i2group.disco.user.WebSphereUserGroupsProvider`

By default, the value is `com.i2group.disco.user.WebSphereUserGroupsProvider`

DynamicSecuritySchemaResource

The security schema file, which defines the security dimensions and values that are available in the system, and the mapping of user groups to dimension values. This (XML) file needs to be on the classpath.

By default, the value is `example-dynamic-security-schema.xml`

AuditLogger

The full class name of the audit logger. If no auditing is required, specify `com.i2group.disco.audit.NoOpAuditLogger`

By default, the value is `com.i2group.disco.audit.NoOpAuditLogger`

DefaultSecurityDimensionValuesProvider

The full class name of the default security dimension values provider. Use `com.i2group.disco.user.PropertyBasedDefaultSecurityDimensionValuesProvider` to set the default security dimension values on a record to the list from the `DefaultSecurityDimensionValues` element in your security schema. This is only applicable to the Information Store with Opal Services.

By default, the value is `com.i2group.disco.user.PropertyBasedDefaultSecurityDimensionValuesProvider`

The DiscoServerSettingsCommon.properties file

The following properties are in the `DiscoServerSettingsCommon.properties` file:

IdleLogoutMinutes

The idle time in minutes after which the end user is logged out.

By default, the value is 15

AlertBeforeIdleLogoutMinutes

The time in minutes the end user is alerted prior to being logged out due to inactivity.

By default, the value is 2

AlwaysAllowLogout

Forces logout to always be available for all authentication methods.

By default, the value is false

DeletedChartTimeoutSeconds

The time in seconds till a deleted chart is permanently removed.

By default, the value is 7200

ResultsConfigurationResource

The file that specifies what options are available to users when they view and filter results.

CommandAccessControlResource

The file that is used to specify specific group access to commands.

EnableSolrIndexScheduler

Turn on or off the scheduling of indexing Setting this option to false disables the scheduler and should be used when ingesting large amounts of data.

By default, the value is true

SolrHealthCheckIntervalInSeconds

The interval time in seconds between the checks of the Solr cluster status.

By default, the value is 60

QueryTimeoutSeconds

The time in seconds after which the server can cancel a search, resulting in an error. This is not an absolute limit. A search might continue to run for several seconds after the limit is reached, but it should terminate within a reasonable time. A zero setting disables search timeout.

By default, the value is 60

WildcardMinCharsWithAsterisk

The minimum number of characters (not counting asterisks) that must be provided in a wildcard text search query that contains asterisks.

By default, the value is 0

WildcardMinCharsWithQuestionMark

The minimum number of characters (not counting question marks or asterisks) that must be provided in a wildcard text search query that contains question marks.

By default, the value is 0

SearchTermMaxLevenshteinDistance

The maximum Levenshtein distance for spelled-like text searches. The allowed values are 2, 1, and 0. Set the value to 0 to turn off spelled-like text searches.

By default, the value is 2

VisualQueryWildcardMinCharsWithAsterisk

The minimum number of characters (not counting asterisks) that must be provided in a visual query condition that contains or implies asterisks. (Matches wildcard pattern; Starts with; Ends with; Contains)

By default, the value is 0

VisualQueryWildcardMinCharsWithQuestionMark

The minimum number of characters (not counting question marks or asterisks) that must be provided in a visual query condition that contains question marks. (Matches wildcard pattern)

By default, the value is 0

VisualQueryConfigurationResource

The file that specifies what operators are valid in visual query conditions that involve particular property types of particular item types.

VisualQueryMaxValuesInList

The maximum number of values in the value list of a visual query condition.

By default, the value is 10000

RecordMaxNotes

The maximum number of notes that can be added to a record.

By default, the value is 50

MaxRecordsPerDeleteRequest

The maximum number of records that can be deleted in one request.

By default, the value is 500

MaxSourceIdentifiersPerRecord

The maximum number of source identifiers that can be present on a record.

By default, the value is 50

MaxSeedsForDaodServices

The maximum number of records to send as seeds to any i2 Connect service in a "DAOD" or "combined" deployment.

By default, the value is 500

AlertScheduleTimeOfDay

The time of day to run the Visual Queries that are saved with alerting enabled. The format is HH:mm, HH is the hour of the day in a 24-hour time format, 00-23, and mm is the number of minutes past the hour, 00-59.

By default, the value is 00:00

ExpandMaxResultsSoftLimit

The soft limit for the maximum number of results that can be returned for an Expand operation.

By default, the value is 0

SourceReferenceSchemaResource

The file that restricts the source names and types that users can specify in source references when creating and importing records.

LinkMatchEndRecordCombinationLimit

The maximum number of link end combinations that are searched for when performing link matching.

By default, the value is 100

ChartUnsavedChangesLifespanDays

The length of time for which the server retains unsaved changes to a chart across user sessions. Any change to an unsaved chart resets the timer.

By default, the value is 7

ChartUnsavedChangesCleanupScheduleExpression

The schedule on which to clean up unsaved changes to charts when ChartUnsavedChangesLifespanDays is exceeded. The format is a Unix C cron expression.

By default, the value is 0 0 * * 0

Redeploying and resetting i2 Analyze

Most changes to the configuration of i2 Analyze require you to redeploy the system. During development, it is common to clear data from Information Store, or to remove the Information Store and re-create it.

Significant changes to the Information Store schema or the security schema of an i2 Analyze deployment usually invalidate the information in the Information Store, or even the store itself. As you develop your schemas, you often need either to remove and re-ingest your test data, or to remove and re-create the store.

After any change to the toolkit configuration of a deployment of i2 Analyze, you must redeploy in order to update to the running deployment.

Performing these actions in sequence is common before in your development environments, but rare in production. If you do run them in sequence, the combination effectively resets the system to its just-deployed state.

You might also need to reset a deployment of i2 Analyze to its just-installed state. For example, if you completed an example deployment and then want to start developing a production deployment on the same hardware.

Clearing data from the system

The `clearData` task can remove data that is stored in both the search index and the Information Store database. You can use this task to remove test data from a system.

In a multiple server deployment, run the `clearData` and `clearSearchIndex` tasks from the Liberty server.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `clearData` and `clearSearchIndex` on the Liberty server that you used to create the database only.

Important: These instructions are designed to remove data that is used for test purposes. Do not use the task to clear data in a production system without backing up your data. For more information about backing up your data, see [Backing up a deployment](#).

1. On the Liberty server, in a command prompt navigate to the `toolkit\scripts` directory.
2. Stop the deployment:

```
setup -t stopLiberty
```

3. To clear data and the search index, run the following command:

```
setup -t clearData --hostname liberty.host-name
```

- Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.
- A message is displayed when you run the task to confirm that you want to complete the action. Enter `Y` to continue. The data and the search index are removed from the system.

If you run the `clearData` task with the `--scripts` argument, the scripts for clearing the Information Store database are generated in the `toolkit\scripts\database\<database dialect>\InfoStore\generated\clearData` directory, but not run. You can then inspect the scripts before they are run, or run the scripts yourself. To run the scripts yourself, run them in their numbered order to clear the data from your system.

To generate the scripts for the task, run:

```
setup -t clearData --scripts
```

After you run the scripts manually, clear the search index:

```
setup -t clearSearchIndex --hostname liberty.host-name
```

4. Start the deployment of i2 Analyze.

```
setup -t startLiberty
```

Resetting the system

The `dropDatabases` task can remove not only the data, but also the underlying structures such as tables from your database management system. The `deleteSolrCollections` task can remove any Solr collections from the system.

To drop the databases, you must ensure that there are no active connections to them.

In a multiple-server environment, run all toolkit tasks from the Liberty server.

In a deployment that provides high availability, use the documentation from your database management system to remove the Information Store database from each database server in your deployment instead of the `dropDatabases` toolkit task.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `clearSearchIndex` on the Liberty server that you used to create the database only.

Important: These instructions are designed to remove databases that are used for test purposes. Do not use the task to remove databases in a production system without backing up your data. For more information about backing up your data, see [Backing up a deployment](#).

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop the deployment:

```
setup -t stopLiberty
```

3. To remove the database and Solr collections, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -t dropDatabases --hostname liberty.host-name  
setup -t deleteSolrCollections --hostname liberty.host-name
```

Here, *liberty.hostname* is the hostname of the Liberty server where you are running the command. It matches the value for the *host-name* attribute of the `<application>` element in the `topology.xml` file.

A message is displayed when you run each task to confirm that you want to complete the action. Enter *Y* to continue. The database and Solr collections are removed from the system.

4. To re-create the Solr collections and databases, run the following commands:

```
setup -t createSolrCollections --hostname liberty.host-name  
setup -t createDatabases
```

5. Start the deployment of i2 Analyze.

```
setup -t startLiberty
```

Stopping and starting i2 Analyze

You might need to stop and start components of your i2 Analyze deployment to complete some configuration changes.

If you are using the single-server or remote database deployment topologies, you can use the `stop` and `start` toolkit tasks to stop and start i2 Analyze.

If you are using the multiple-server deployment topology, you must stop and start the components of i2 Analyze individually.

1. Stop the components of i2 Analyze.

- a) Stop ZooKeeper.

To stop ZooKeeper, run the following command on every server where ZooKeeper is running:

```
setup -t stopZkHosts --hostname zookeeper.host-name
```

Here, *zookeeper.host-name* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the *host-name* attribute of a `<zookeeper>` element in the `topology.xml` file.

- b) Stop Solr.

To stop Solr, run the following command on every server where Solr is running:

```
setup -t stopSolrNodes --hostname solr.host-name
```

Here, *solr.host-name* is the hostname of the Solr server where you are running the command, and matches the value for the *host-name* attribute of a `<solr-node>` element in the `topology.xml` file.

- c) Stop Liberty and the i2 Analyze application.

To stop Liberty, run the following command on each Liberty server:

```
setup -t stopLiberty
```

2. Start the components of i2 Analyze.

- a) Start ZooKeeper.

To start ZooKeeper, run the following command on every server where your ZooKeeper hosts are located:

```
setup -t startZkHosts --hostname zookeeper.host-name
```

Here, *zookeeper.host-name* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the *host-name* attribute of a `<zkhst>` element in the `topology.xml` file.

b) Start Solr.

To start Solr, run the following command on every server where your Solr nodes are located:

```
setup -t startSolrNodes --hostname solr.host-name
```

Here, *solr.host-name* is the hostname of the Solr server where you are running the command, and matches the value for the *host-name* attribute of a `<solr-node>` element in the `topology.xml` file.

c) Start Liberty and the i2 Analyze application.

To start Liberty, run the following command on each Liberty server:

```
setup -t startLiberty
```

Redeploying Liberty

Redeploy Liberty to update the i2 Analyze application with your configuration changes.

In a multiple-server environment, run all toolkit tasks from the Liberty server.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

```
setup -t stopLiberty
```

3. Update the i2 Analyze application:

```
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

5. If you are using the IBM HTTP Server, start or restart it.

Setting up your XSD aware XML editor

Some configuration files are provided with an associated XSD file. The XSD file provides client-side validation, and makes it easier to develop your XML configuration files.

It is recommended that you use an XSD-aware XML editor with intelligent code completion.

The following instructions describe how to install Visual Studio Code and the XML Language Support by Red Hat extension.

1. Download and install Visual Studio Code, <https://code.visualstudio.com/>.

2. In Visual Studio Code, install the XML Language Support by Red Hat extension.

For information about how to install extensions, see <https://code.visualstudio.com/docs/editor/extension-gallery>.

After you install and configure your editor, you should use this editor whenever the instructions tell you to use an XSD-aware XML editor.

If you are using Visual Studio Code, the associated XSD file must be in the same directory as the XML file that you are editing.

Using the admin endpoints

The reload endpoints support the POST method, which you must call through a command-line tool such as **postman** or **curl**. The endpoint requires authentication, which means that you must log in to the server as an administrator and retrieve a cookie before you can POST to reload.

To use the admin endpoints, you must log in as a user that has the administrator permission. To provide a user with the `i2:Administrator` permission, use command access control to specify the user group to give access to the administrator role. For more information, see [“Configuring command access control”](#) on page 143.

The following steps demonstrate how to use **curl** to POST to the admin endpoints.

You can find the REST API reference documentation at the following URL on the i2 Analyze server: http://host_name/conext_root/doc. For example, <http://localhost/opa1/doc>.

1. If the **curl** utility is not available on your server, download it from the project website at <https://curl.haxx.se/download.html>.
2. Open a command prompt and use **curl** to log in to the i2 Analyze server and retrieve an authorization cookie:

```
curl -i --cookie-jar cookie.txt
-d j_username=user_name
-d j_password=password
http://host_name/conext_root/j_security_check
```

This command connects to the specified i2 Analyze server as the specified user, retrieves the authentication cookie, and saves it to a local file named `cookie.txt`. The LTPA token in the cookie is valid for 2 hours.

When the command completes, the response from the request is displayed. If the retrieval is successful, the first line of the response is:

```
HTTP/1.1 302 Found
```

After you retrieve and save the cookie, you can POST to the admin endpoints. When you POST to the admin endpoints, include the cookie file that you received when you logged in.

3. The following command is an example of a POST to the admin endpoint to reload the live configuration:

```
curl -i --cookie cookie.txt
-X POST
http://host_name/conext_root/api/v1/admin/config/reload
```

When the command completes, the response from the request is displayed. If the retrieval is successful, the first line of the response is:

```
HTTP/1.1 200 OK
```

If you see the HTTP/1.1 403 Forbidden response code, the user for which you retrieved the token does not have the command access control administrator permission, or the token has expired:

- If your token has expired, you can get a new one by running the command in step 2 again.
- If your user does not the command access control `i2:Administrator` permission, see [“Configuring command access control” on page 143](#) to provide the user with the permission.



Warning: The `reload` method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

Connecting to external data sources

Subject to your licensing terms, a deployment of i2 Analyze can use the i2 Connect gateway to query and retrieve data from external data sources. By implementing a connector to an external data source, you enable i2 Analyze to create and display records that represent the data in that source.

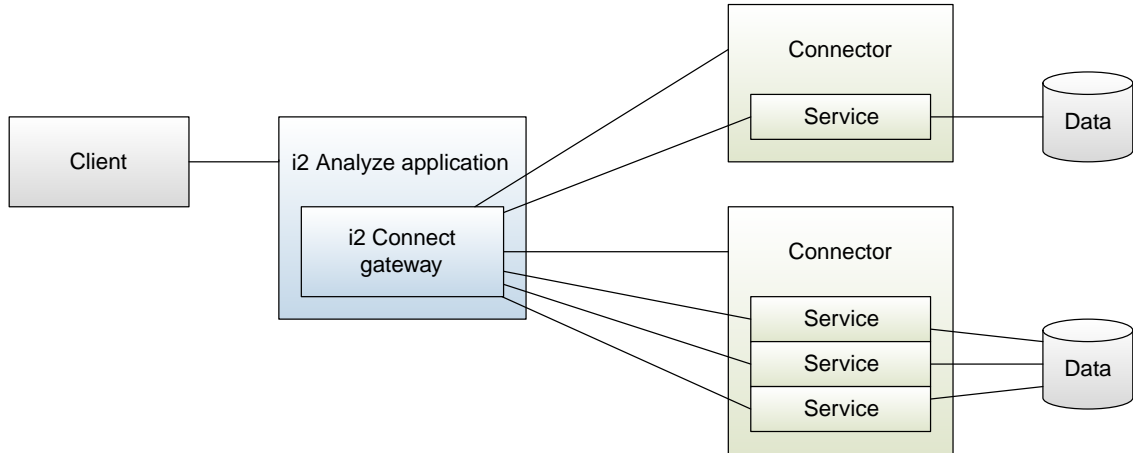
If the deployment and their permissions support it, users who create records from external sources can upload them to the Information Store for storage, sharing, and subsequent analysis.

Solutions that use the i2 Connect gateway depend on *connectors* that you create to make the bridge between i2 Analyze and external data sources. Conventionally, you write one connector for each source that you want to connect to. Each connector provides one or more *services* that present a querying interface to users and perform the actual queries on the data source.

i2 Analyze and the i2 Connect gateway

In an i2 Analyze deployment, you can provide users with access to an Information Store, or enable them to query data in one or more external sources, or both. i2 Analyze deployments that are targeted at external data access include the i2 Connect gateway alongside or instead of the Information Store.

In a complete solution, the i2 Connect gateway provides i2 Analyze with information about what connectors and services are available and how clients are to use them. i2 Analyze passes client requests for data to the gateway, which receives and responds to the requests on behalf of those services.



Each connector contains the code that interacts with an external data source, the definitions of the services that it supports, and the descriptions of how clients might present its services to users. For example, clients need to know the following information:

- The name and description of each service a user has access to
- Whether a service requires users to authenticate before they can use it
- Whether a service supports parameters that users can specify when they run it
- The input controls to display, and the validation to perform, for each parameter
- Whether a service behaves differently as a result of the current chart selection (any selected records are then *seeds* for the operation that the service performs)

One of the roles of the gateway is to retrieve this information from all the connectors that i2 Analyze knows about. It can then help to ensure that requests to and responses from services are formatted correctly.

In solutions like these, a connector is an implementation of a REST interface that enables the gateway to perform its role. More specifically, a connector must support the following tasks:

- Respond to requests from the gateway for information about its services
- Validate and run queries against an external data source, including any queries that require parameters or seeds
- Convert query results to a shape that the gateway can process into i2 Analyze records

Writing a connector requires you to develop an implementation of the REST interface that the gateway expects, and to write queries that retrieve data from an external source. You must also understand the i2 Analyze data model, and be able to convert the retrieved data so that it conforms to an i2 Analyze schema.

The examples that IBM provides in the open-source repository at <https://github.com/ibm-i2/analyze-connect> demonstrate meeting several of these requirements in fully functional connectors. The remainder of this section explains how to meet them in more detail.

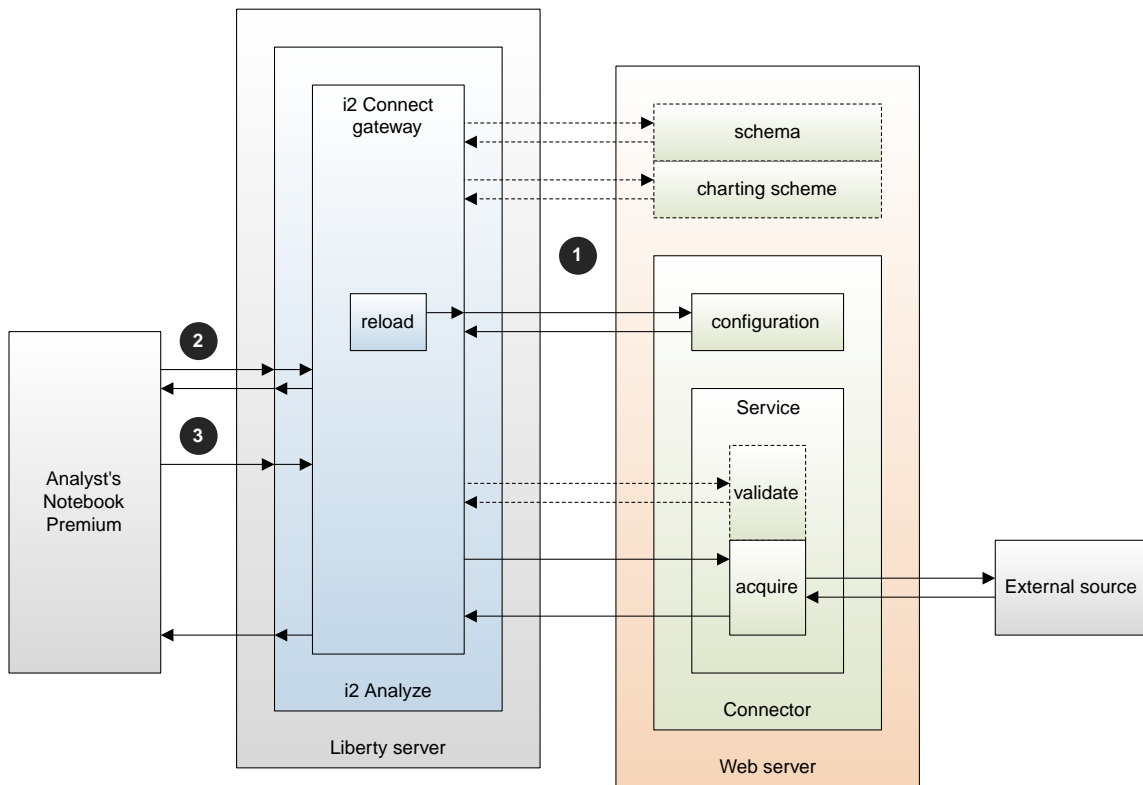
Note: For information on securing the communication between the i2 Connect gateway and its connectors, see [Client authenticated Secure Sockets Layer](#) and the [i2 Connect gateway](#).

System architecture with the i2 Connect gateway

The simplest possible deployment of i2 Analyze that uses connectors to access an external data source has four parts. The client, in this case Analyst's Notebook Premium, communicates with a deployment of i2 Analyze that includes the i2 Connect gateway. The gateway manages at least one connector, which in turn exchanges data with an external source.

System interactions

The following diagram represents the deployment that is described in the introduction, and the interactions that take place between each part.



The diagram also identifies five of the REST endpoints that connectors can implement and use:

Configuration endpoint

The gateway sends a request to the mandatory configuration endpoint to gather information about the services that the connector supports. All connectors have at least one service, and it is a service that implements the validate and acquire endpoints.

Note: For connectors whose services present modified behavior to different users, the information is split between two endpoints: this one, and the user-specific configuration endpoint.

Schema endpoint

The schema endpoint is optional, and so is where you implement it. If a connector returns results whose types are not in the Information Store schema or a gateway schema, you can provide

a connector schema that define those types. The configuration specifies whether and where a connector schema is available.

Charting scheme endpoint

The charting scheme endpoint is optional, and closely associated with the schema endpoint. A connector configuration that specifies a schema endpoint also specifies a charting scheme endpoint.

Validate endpoint

The validate endpoint is optional. If the configuration endpoint says that a particular service supports it, then the gateway sends a request to the validation endpoint immediately before a request to the acquire endpoint. The purpose of the validate endpoint is to test whether a request is reasonable before it is run.

Acquire endpoint

The acquire endpoint is mandatory for all services that provide synchronous querying. During a synchronous query, the gateway sends a request to the acquire endpoint that tells the service to query the external source. The service receives data back from the source and places it into the response that it returns to the gateway.

Note: For services that provide asynchronous queries, the query resource endpoint is mandatory instead of the acquire endpoint.

Interaction sequence

The interaction between the i2 Connect gateway and the other parts of an i2 Analyze deployment takes place in three distinct phases, labeled 1, 2, and 3 in the diagram.

1. When i2 Analyze starts up, the i2 Connect gateway sends a request to the configuration endpoint of every connector that is listed in the topology. It caches the information that it receives in response.

If the configuration for any of the connectors specifies schema and charting scheme endpoints, the gateway sends requests to retrieve the connector schemas and charting schemes from those endpoints.

Note: You can also force the gateway to repeat this process while i2 Analyze is running. The gateway implements [the reload endpoint](#), whose POST method you can call after you change the configuration of a connector.

2. When a client connects to i2 Analyze, the latter gets the cached information from the gateway and returns it to the client. The client can then present the queries that the services implement to users, and help users to provide valid parameters and seeds to those queries where appropriate.

Note: For connectors that implement the user-specific configuration endpoint, i2 Analyze sends a request to it with information about the connecting user at this stage, to retrieve the remainder of the configuration information.

3. When a user runs a synchronous query, i2 Analyze passes it to the gateway for processing. The gateway packages the query into a request to the acquire endpoint on the service in question, preceded by a request to the validate endpoint if the service supports it.

On receiving the response from the acquire endpoint, the gateway converts the data that it contains into results that i2 Analyze can ultimately return to the client.

System security with the i2 Connect gateway

You can configure three different aspects of security in a deployment of i2 Analyze with the i2 Connect gateway. You can secure the connection between the gateway and a connector; you can restrict

which i2 Analyze users can run the queries that connectors implement; and you can write your connectors so that users must authenticate before they can run queries.

Secure the connection to a connector

You can use client-authenticated SSL communication to secure the connection between the i2 Connect gateway and a connector. For more information, see [Client-authenticated Secure Sockets Layer with IBM i2 Connect](#).

Restrict access to a connector

You can use the command access control feature to control which users can use a connector. Through user groups, you can prevent users from using the i2 Connect gateway altogether, or you can restrict them to using only a subset of the deployed connectors. For more information, see [Controlling access to features](#).

Require users to authenticate before running queries

You can write a connector so that users must authenticate before they can use one or more of the services that the connector provides. This feature enables connectors to interact with data sources that require users to log in, or to request authentication for their own purposes. For more information, see [Supporting authentication](#).

As well as these specific applications of i2 Analyze security, there are two other mechanisms that can affect which services a user has access to:

- When a user is prevented from seeing all records of a particular type, any service that requires seeds or returns results of that type is not visible to that user.
- A connector that supports user-specific configuration can, if it chooses, provide a different list of available services to different users.

Creating a connector to an external data source

To create a connector, you need a data source to query, and a server that supports REST endpoints to query it from. On that foundation, you can build the functionality that takes requests from the i2 Connect gateway, retrieves data from the source, and returns it to the gateway in a shape that is ready for conversion to i2 Analyze records.

The easiest approach to writing a connector is to start by transferring data from the external source to the i2 Analyze server - and then to the client - with as few complications as possible. When you have that mechanism, you can use it as a base from which to develop the more complex services that you want to implement.

Note: The example connectors at <https://github.com/ibm-i2/analyze-connect> contain services that between them demonstrate several aspects of connector development. As you read this documentation, it is helpful also to read the source code for those examples.

Understanding the external source

Data is only valuable in i2 Analyze if users can examine it alongside, and with the same tools as, all the other data in the system. When you connect to an external source, you must decide how its contents relate to existing data and plan your connector development accordingly.

Before you write a connector, it is helpful to understand:

- What data is available from the external source
- How the available data maps to any schemas that you already have
- What your plans are for the finished connector

For example, if the purpose of your connector is to provide data from an external source to an existing deployment, it might be that some or all of the data in the source is similar to data that the deployment already models.

Alternatively, if you are writing a connector for use in many deployments, there are advantages to providing a description of the data that the connector returns alongside the connector itself.

During connector development, you can decide whether the definition for each type that you want to return is found in the Information Store schema, a gateway schema, or the connector schema. In the first two cases, you can also decide whether to use an item type that already exists or to extend the schema with a new type.

1. Consider the type of the source and how you can query it.

For spreadsheets and text files, it is likely that you have complete access to the data they contain. For databases and web services, you might be restricted to a set of predefined queries.

2. Identify the types of data that you can retrieve from the source. If your aim is to integrate with an existing deployment, compare the source's types with the types in the Information Store or gateway schemas.

For each type in the source, you can reach one of three conclusions:

- The type and its data are a match for (or a subset of) one of the entity or link types you already have
- The type is a match for one of the types you already have, but its data implies new property types
- The type is not currently represented in the existing schemas

3. If the Information Store schema or a gateway schema already describes all the data in the external source, then you can start the process of writing queries that retrieve data from that source.

Note: In a deployment that contains more than one gateway schema, any single connector can use the types from only one gateway schema.

4. If the data in the external source contains *property* types that the existing schemas do not describe, then you can ignore that data or edit a schema to add those property types.

Additive changes to schemas are always permitted.

5. If the data in the external source contains *item* types that the existing schemas do not describe, then you can use one of the techniques for adding them to a schema, or create a connector schema to describe them.

Adding types to an i2 Analyze deployment

The data that you retrieve from an external source through the i2 Connect gateway must have entity and link types that are defined somewhere in the i2 Analyze deployment. If none of the existing schemas contains types that correspond with the external data, then you must extend or write a new schema as part of connector development.

If the data that you can retrieve from the external source is compatible with existing schema types, and your connector does not need to define its own types, then you can skip this task. Otherwise, you need to add type definitions to the i2 Analyze deployment.

There are two ways to add types to an existing deployment of i2 Analyze:

- Add the new entity, link, and property types directly to one of the deployed schemas in the usual way. Additive changes to schemas are always allowed.
- Create a gateway or connector schema (along with an accompanying charting scheme), and define the new item types in that schema.

Choosing which approach to use depends on your aims:

- Adding types to the Information Store schema makes them immediately and permanently available throughout the i2 Analyze deployment. Users can upload records that they retrieve from the external source to the Information Store without any further intervention.
- Especially in a deployment that does not include the Information Store, adding types to an existing gateway schema has some of the same benefits. The new types appear at familiar locations in the Analyst's Notebook Premium user interface, and they become available to future connectors.
- A key reason for creating a gateway schema to contain your new types is when you are creating a library of connectors that provide access to a particular source. The types in the gateway schema can be shared by all the connectors, and you can provide the gateway schema alongside the connectors if you need to distribute them.
- Creating a connector schema provides the highest level of flexibility. You can add the connector with its schema to any i2 Analyze deployment that includes the i2 Connect gateway. And you can make modifications to the connector schema without affecting the other schemas. But by its nature, this approach is also the least well integrated with the rest of the deployment.
- If you decide to extend one of the deployed schemas, you can do so in the same way that you would add types for any other reason:
 - a) Open the XML file that contains the deployed schema in Schema Designer.
 - b) Add entity and link types (or property types) to the schema that correspond to data from the external source.
 - c) Add configuration for new and modified entity and link types to the charting scheme.
 - d) Update the deployment with the modified schema and charting scheme.

For more information, see the [Schema Designer user guide](#), and the documentation on [modifying the Information Store schema](#) and [modifying gateway schemas](#).
- If you decide to create a schema, then you still use Schema Designer, but the process is different:
 - a) In Schema Designer, select **File > New Schema**.
 - b) Edit the schema and its charting scheme to add types for the data from the external source, and then save and close the XML files.
 - c) To deploy a gateway schema, make it (and its charting scheme) available to the deployment toolkit, and then follow the instructions in [Adding, removing, and modifying gateway schemas](#).
 - d) To deploy a connector schema, arrange for it (and its charting scheme) to be available from endpoints on the server that hosts the connector.

To complete this approach, you must tell the i2 Connect gateway where to find the endpoints that you created. You provide that information by [adding to your implementation of the configuration endpoint](#).

Writing a query with no parameters

All connectors contain at least one *service* that retrieves information from an external source. By starting with a service that runs a simple query, you can focus on aspects such as the REST endpoint implementations and data structures that are common to all services.

As you create a connector to an external source, writing a query that does not require parameters makes early testing and development easier. Not only does it simplify the connector, but also it simplifies running the query from the client.

No fixed set of steps for retrieving data from external data sources exists, but some of your choices can make later tasks easier. Whatever the source, your initial aim is to generate a small set of representative data.

- If the source allows it, begin by writing a tightly bound query. Or, if your source comprises one or more text files, try to abbreviate them.

At the start of the process, large volumes of data can be distracting.

- If you can, make the query retrieve data for entities *and* links.
Creating the data structures that represent linked entities is an important part of connector development.
- Try to generate a data set that has most of the types that the external source contains.
No matter how complex your queries eventually become, the code to process data into the right shape for i2 Analyze is unlikely to change.

When you implement an `acquire` endpoint, you can call the query that you developed here directly from that code. Alternatively, during development, you might decide to run the query now, and save the results in a file.

Implementing a configuration endpoint

The configuration endpoint of a connector must respond to requests from the i2 Connect gateway with information about the schemas that the connector uses and the services that it provides. In some more complex connectors, the configuration endpoint can indicate in its response that more information is available from other endpoints.

Navigate to the source code for any of the connector projects in the connector directory of the `analyze-connect` repository. In a text editor, open one of the `config.json` files that implements the configuration endpoint and demonstrates an approach that you might take.

When the i2 Analyze server starts, the i2 Connect gateway sends a request to the configuration endpoint. The response that your connector sends back provides information to the gateway that includes a list of what services are available. i2 Analyze caches this information and provides it to clients when they connect.

Note: Some connectors support user-specific configuration, where different users see different views of the services that the connector exposes. In those circumstances, a static list of services is not appropriate. Instead, the configuration endpoint provides the location of the *user configuration endpoint*, which provides per-user information about the connector's services.

The simplest possible service does not behave differently for different users, does not support seeds, and does not require users to provide parameters. As a result, it does not require the client to display user input controls. In that case, the structure that you return from the configuration endpoint must contain only two objects:

- The `defaultValues` object.

Within the `defaultValues` object, you must specify the time zone for the i2 Connect gateway to apply to any retrieved value that does not specify its own time zone. There are also some optional settings:

- You can specify a direction for any retrieved link that does not specify its own direction.
- You can specify the identifiers and locations of the entity and link types that the i2 Connect gateway applies to records if a service does not indicate the types of data that it can retrieve.
- You can provide a default setting for whether the data identifiers that the connector returns are persistent from one call to the next.

Note: If you do not indicate whether these identifiers are persistent here or in the `services` array, the gateway assumes that the identifiers are *not* persistent.

- The `services` array.

The `services` array must contain information for at least one synchronous or asynchronous service. The information must include a unique identifier and a name for the service. It must also specify whether the service requires a client UI, and the URL of the `acquire` or `queriesResource` endpoint.

First, construct the `defaultValues` object:

1. Determine the time zone that is most likely to apply to any temporal data in the source that the services in this connector query.
2. Find the identifier of the time zone in the IANA Time Zone Database, and arrange for the response from the endpoint to include the identifier in its `defaultValues` object.

For example:

```
{
  "defaultValues": {
    "timeZoneId": "Europe/London",
    ...
  },
  "services": [
    ...
  ]
}
```

Note: You can also retrieve a list of supported time zones from the GET method on the i2 Analyze server's `/api/v1/core/temporal/timezones` REST endpoint.

3. If the source contains only a few types, and if you intend the connector eventually to have many services that run different queries, then consider adding an `entityTypeId` and a `linkTypeId` to the `defaultValues` object.

It is more common to specify what types of record a query might retrieve on a per-service than on a connector-wide basis. However, if you do not supply default types here, then every service must supply types individually.

The gateway searches for the types that you supply here in the connector schema, the gateway schema, and the Information Store schema. To specify exactly where the types are defined, you can add `entityTypeLocation` and `linkTypeLocation` properties.

4. If you know that the source will always attach the same identifier to the same piece of retrieved information, add the `resultIdsPersistent` field to the `defaultValues` object and set its value to `true`.

Note: The `resultIdsPersistent` field is also valid at the per-service level, where its value overrides any connector-wide setting.

Then, add an object to the `services` array:

5. To the `services` array, add a service object that has an `id` and a `name`. It is common to include a `description` and populate the `resultItemTypes` array as well, although neither is mandatory.

For example:


```

{
  "defaultValues": {
    "timeZoneId": "Europe/London",
    "resultIdsPersistent": true
  },
  "services": [
    {
      "id": "nypd-service",
      "name": "NYPD Connector: Get All",
      "description": "A service that retrieves all data",
      "resultItemTypeIds": {
        "INFOSTORE": ["ET1", "ET2", "ET3", "LT1", "LT2", "LT3"]
      },
      ...
    }
  ]
}

```

6. For a service whose query does not allow callers to provide parameters, you can set the `clientConfigType` to `NONE`.

```

{
  "defaultValues": {
    "timeZoneId": "Europe/London",
    "resultIdsPersistent": true
  },
  "services": [
    {
      "id": "nypd-service",
      "name": "NYPD Connector: Get All",
      "description": "A service that retrieves all data",
      "resultItemTypeIds": {
        "INFOSTORE": ["ET1", "ET2", "ET3", "LT1", "LT2", "LT3"]
      },
      "clientConfigType": "NONE",
      ...
    }
  ]
}

```

If you later add parameters to the query, you can allow users to specify them by changing the value to `FORM` and providing the identifier of a client configuration in `clientConfigId`.

If you later add support for seeds, you must add a `seedConstraints` object to the service.

7. Finally, for a synchronous service, set the `acquireUrl` of the service to the URL where you intend to host the acquire endpoint.

```

{
  "defaultValues": {
    "timeZoneId": "Europe/London",
    "resultIdsPersistent": true
  },
  "services": [
    {
      "id": "nypd-service",
      "name": "NYPD Connector: Get All",
      "description": "A service that retrieves all data",
      "resultItemTypeIds": {
        "INFOSTORE": ["ET1", "ET2", "ET3", "LT1", "LT2", "LT3"]
      },
      "clientConfigType": "NONE",
      "acquireUrl": "/all"
    }
  ]
}

```

This JSON structure represents (almost) the simplest response that you can send from the configuration endpoint of a connector. It contains the definition of one simple service. In order for the i2 Connect gateway to retrieve that definition, you must add details of the connector to the i2 Analyze deployment topology.

Adding a connector to the topology

When you have an implementation of a configuration endpoint, you can tell i2 Analyze about the connector that it describes by adding it to the topology file. In that file, the `<connector>` element controls the behavior of the connector and the experience of its users.

Ensure that your deployment of i2 Analyze includes the i2 Connect gateway. If the gateway was not a part of the original deployment, you can add it by following the instructions in [Adding more data sources](#).

Each `<connector>` element in the topology file ties together the URL of a server, which might host several connectors, and the URL of the configuration endpoint for a particular connector. After you change the contents of the topology file, you must update the deployment and restart the i2 Analyze server.

When you deploy i2 Analyze with one of the example configurations that includes the i2 Connect gateway, the topology file contains a `<connectors>` element. Adding a connector to the topology means adding a child to that element. (If your topology file does not contain a `<connectors>` element, you can also add it by hand.)

1. Navigate to the `configuration\environment` directory of the toolkit for your deployment of i2 Analyze. In a text editor, open the `topology.xml` file.
2. In the `<war>` element, locate the `<connector-ids>` child element, or add it if necessary.
3. Add a `<connector-id>` child element with the following format:

```

<connector-ids>
  <connector-id value="ConnectorId"/>
</connector-ids>

```

Here, *ConnectorId* must be unique within the topology file. Its purpose is to reference the configuration for the connector in the <connectors> element.

4. Add a <connector> element to the <connectors> element, which is a child of the root <topology> element:

```
<connectors>
  <connector id="ConnectorId"
            name="ConnectorName"
            base-url="Protocol://HostName:PortNumber"
            configuration-url="Protocol://HostName:PortNumber/Path"/>
</connectors>
```

Here, *ConnectorName* is likely to be displayed to users in the list of queries that they can run against external sources.

The i2 Connect gateway uses the value that you assign to the *base-url* attribute as the stem for the URLs that you specify in the configuration (such as */exampleSearch/acquire*). *configuration-url* is the only optional attribute.

By default, the gateway attempts to retrieve the configuration from *<base-url>/config*. You can change this behavior by specifying a different URL as the value of *configuration-url*.

5. Save the file, and then update the deployed connectors configuration and restart the i2 Analyze server.
 - a) On the command line, navigate to `toolkit\scripts`.
 - b) Run the following commands in sequence:

```
setup -t stopLiberty
setup -t updateConnectorsConfiguration
setup -t startLiberty
```

When the server restarts, the i2 Connect gateway makes its request to the configuration endpoint. If that endpoint (or any of the endpoints in the retrieved configuration) is not available, the result is not an unrecoverable error. Rather, the i2 Analyze server logs the problem, and users see messages about unavailable services in the client application.

To preview what users see, and to discover the status of the configured connectors at any time, you can use the server admin console.

6. Open a web browser and connect to the i2 Analyze server admin console at `http://host_name/opa1/admin`.

The admin console displays a list of connectors with their statuses. To see what the logged-in user would see in their client application, click **Preview services**.

Implementing an acquire endpoint

In a connector, an acquire endpoint must perform two tasks in succession. Its roles are to retrieve data from an external source, and to return that data to the i2 Connect gateway in a form that the latter can use to create i2 Analyze records.

Refer again to the example connector projects at <https://github.com/ibm-i2/analyze-connect>. They contain implementations of the acquire endpoint for several services, which cover many of the common considerations.

When a user runs a synchronous query against an external data source through their client application, the i2 Connect gateway makes a request to the acquire endpoint of the relevant service.

The request always has a *payload* that can contain information (often, parameter values or seed records) that the user provided to the query.

In general, your implementation of an acquire endpoint uses the contents of the payload to refine the commands that it sends to the external source. If the service that owns the endpoint specified a `clientConfigType` of `FORM`, then the identifiers of conditions in the form match the identifiers of conditions in the payload. If the service specified `seedConstraints`, then the payload contains the property values of seed records.

For a query with no parameters and no seeds, the payload is empty. You can focus your development effort on making sure that the response from the acquire endpoint meets the requirements of the i2 Connect gateway.

Note: It is not possible to follow the same procedure for every different kind of external data source. The following steps do not apply in all cases. Rather, they describe tasks that you are likely to require.

1. In IBM i2 Analyze Schema Designer, open the schema file that contains the definitions of the i2 Analyze entity, link, and property types that data from the external source must align with.
2. Make a note of the identifiers and logical types of the property types of each of the entity and link types in the retrieved data. Make sure that you know which property types are mandatory.
3. Run the command to retrieve data from the external data source:
 - a) If data retrieval from the external source succeeds, store the data in a structure that you can manipulate easily, and continue with the next step in this procedure.
 - b) If data retrieval from the external source fails, add the optional `errorMessage` field to the response from the endpoint.

The `errorMessage` field contains a string that users of the connector see when a problem occurs. When the field is present (and not empty), the operation fails and any data in the arrays of entities and links is ignored.

4. If the data from the external source contains information about links between entities, but the links in the data have no natural identifiers, you must manufacture those identifiers.

This problem is similar to the one that you face if you [prepare data for ingestion](#) into the Information Store.
5. Create the arrays of entity and link data objects that must appear in the response from the acquire endpoint.
6. Populate the arrays according to the descriptions in the SPI documentation, taking care to ensure that you provide values for all mandatory properties.

Modifying and testing a connector

A connector to an external data source is defined by the information that the i2 Connect gateway retrieves from its configuration endpoints. To add a connector, or when you modify an existing one, you must arrange for the gateway to reload the information for all the connectors in the topology.

A connector that has only a configuration endpoint and an acquire endpoint is simple, but entirely functional. When you have a connector in this state, you can verify that many of the most important mechanisms are working correctly. In general, to enable faster test and development cycles, add new connectors to a running instance of i2 Analyze as soon as you can.

The i2 Connect gateway maintains a cache of configuration information that it provides to clients on request. Depending on the stage of development of your connector, you can use different approaches to make the gateway update its cache:

- If you add or edit a `<connector>` element in the topology file, then you must redeploy and restart the i2 Analyze server, as described in [“Adding a connector to the topology”](#) on page 316.

- If you modify a connector so that the response from its configuration endpoint changes - when you add a `validate endpoint`, for example - you do not need to redeploy the server. Instead, you can either run the `restartLiberty` task from the deployment toolkit, or use the i2 Connect gateway's reload endpoint.

Note:

For a connector that supports user-specific configuration, the i2 Connect gateway maintains per-user caches of configuration information. Using the reload endpoint clears the caches for all users. The per-user caches are not re-created until a user makes a new request to the connector.

Regardless of how you update the gateway's cache of connector configuration information, the effect for clients and their users is the same. When a client next asks for a list of the services that it can use, any changes that you made are reflected in the list.

To run `restartLiberty`, you can use the same technique as for any other deployment toolkit task. This section describes how to use the reload endpoint, and then explains some ways to diagnose problems with a new or updated connector.

The reload endpoint supports the POST method, which you can call through a command-line tool such as **postman** or **curl**. The endpoint requires authentication, which means that you must log in to the server and retrieve a cookie before you can POST to reload.

1. The following steps describe how to use **curl** to reload the connectors:

- a) If the **curl** utility is not available on your server, download it from the project website at <https://curl.haxx.se/download.html>.
- b) Open a command window and use **curl** to log in to the i2 Analyze server and retrieve an authorization cookie:

```
curl -i --cookie-jar cookie.txt
-d j_username=user_name
-d j_password=password
http://host_name:host_port/context_root/j_security_check
```

This command connects to the specified i2 Analyze server as the specified user, retrieves the authentication cookie, and saves it to a local file named `cookie.txt`.

- c) Use **curl** a second time to call the POST method on the reload endpoint:

```
curl -i --cookie cookie.txt
-X POST http://host_name:host_port/context_root/api/v1/connectors/reload
```

This command causes the i2 Connect gateway to reload configuration information from all the connectors that the topology included when it was last deployed. Clients receive updated information about the connectors when they next connect to i2 Analyze.

2. After you reload the connectors (or restart the server), click **Preview services** in the server admin console to check that users will see your changes in their lists of queries.

If you do not see your changes, different symptoms imply different causes:

- If a query does not appear in the list at all, then the problem might lie with either the implementation of the configuration endpoint, or the `<connector>` element in the `topology.xml` file.

If the connector supports user-specific configuration, there might be a problem with the `user` configuration endpoint, or you might be logged in as a user whose configuration does not list the query in question.

- If a query is in the list but marked as unavailable, then the cause of the problem is either the implementation of the acquire endpoint, or the specification of that endpoint in the response from the configuration endpoint.
 - If the displayed information about a query is faulty or incomplete, look again at the definition of the corresponding service in the response from the configuration or user configuration endpoint.
3. Provided that the queries due to new or modified connectors appear correctly, you can open a client, run them, and view the results that they return.

After you test that your connector meets your requirements, you might choose to secure the connector in your deployment. You can control user access to the connector, and secure the connection between the connector and gateway. For more information, see [“System security with the i2 Connect gateway”](#) on page 309.

Extending a system that queries external sources

The simplest possible connector requires only a configuration endpoint and a basic acquire endpoint in order for clients to present its services to their users. By extending these endpoints and adding new ones, you can improve the reliability and utility of the connectors that you write.

Returning source references

Source references can provide users with rich information about where the data they see in i2 Analyze records came from. By adding source references to the record data that a service returns, you enable extra functionality in the client, including the ability to associate and view images with your records.

Source references provide up to five pieces of information about the source of an i2 Analyze record:

- The *type* of the source that data in the record came from
- The *name* of the source
- A *description* of the source
- The *location* of the source, which might be a URL
- An *image* of the source, which might also be an image for the record

When you return source reference information from a service, only the name is mandatory. However, you must also state whether users can edit or delete the source references that you add to your records.

To add source references to the record data that your service returns to clients, you must arrange to include a `sourceReference` object as a peer of the `properties` object in your response from the acquire endpoint.

1. In your code that generates responses from the acquire endpoint, add at least the following extra content:

```

{
  ...
  "properties": {
    ...
  },
  "sourceReference": {
    "source": {
      "name": "source_name"
    }
  },
  ...
}

```

2. To give users the best experience, add a type and a description for your source that align with the definitions in the source reference schema for the deployment:

```

...
"sourceReference": {
  "source": {
    "name": "source_name",
    "type": "source_type",
    "description": "source_description"
  }
},
...

```

3. If the data in the external source has associated images, you can arrange for users to see an image for a record when they view it by adding an image field to the source object:

```

"source": {
  ...
  "image": "image_url"
}

```

You can similarly include the location of the source in the source reference by adding a location field and setting it to either a URL or a text description.

4. By default, when users add records from an external data source to a chart, they cannot edit or delete the source references that you add. To change that, add a userModifiable field to the sourceReference object:

```

...
"sourceReference": {
  "source": {
    ...
  },
  "userModifiable": true
},
...

```

Completing these steps in your service code means that source references are present in the records that users retrieve and view. Source references are optional, so you can control whether to include

them for all records, or just for records of particular types. Finally, source references have the same features in entity and link records, so you do not need to write different code for those two cases.

Using types from different schemas

Simply creating and deploying gateway and connector schemas does not automatically make the item types that they define available to your connectors. When you want to use gateway or connector schemas, you must specify them in the connector configuration, or the i2 Analyze deployment topology, or both.

These instructions assume that you already created gateway and connector schemas that define entity and link types for the data that your connector retrieves, as described in [“Adding types to an i2 Analyze deployment”](#) on page 311.

Anywhere a connector can specify an item type, it can also specify which schema that type is defined in: the Information Store, a gateway, or the connector schema. For the mechanism to work, the connector must provide information in its configuration about the gateway or connector schemas that it wants to use.

Although a deployment of i2 Analyze can support several gateway schemas, an individual connector can use types from only one of them. You choose which gateway schema to use by adding its name to the response from the configuration endpoint.

1. At the top level of the configuration response, add and populate the `gatewaySchema` property.

```
{
  "gatewaySchema": "Example",
  "defaultValues": {
    ...
  },
  "services": [
    ...
  ]
}
```

Provided that a gateway schema with the short name EXAMPLE is configured in the `ApolloServerSettingsMandatory.properties` file for the deployment, the types in that schema are available for use in the connector.

To enable a connector schema, you must add endpoints for retrieving it to any connector that uses it, and then specify the locations of those endpoints in the response from the configuration endpoint.

2. Make your connector schema and its charting scheme available through two simple HTTP GET methods that do not require parameters.

It is easiest, but not mandatory, to implement these endpoints on the server that hosts the connectors that use them.

3. Add `schemaUrl`, `chartingSchemesUrl`, and `schemaShortName` settings to the response from your implementation of the configuration endpoint:


```

{
  "schemaUrl": "/schema",
  "chartingSchemasUrl": "/chartingSchemas",
  "schemaShortName": "Another-Example"
  "defaultValues": {
    ...
  },
  "services": [
    ...
  ]
}

```

In this form, i2 Analyze appends the URLs that you provide to the `base-url` that you specified for the connector in the topology file. By definition, these endpoints are on the same server as the connector. To use endpoints that are implemented elsewhere, you must provide the full URLs here.

Making changes like these affects the connector but not the topology, so you do not need to redeploy i2 Analyze. Restarting the server or using the reload endpoint on the i2 Connect gateway is enough.

4. Follow the steps in [“Modifying and testing a connector”](#) on page 318 to restart the server or force the gateway to reload your connector configuration.

The new types become available for use.

With the above changes in place, and provided that there are no identifier clashes, the connector can use item types from the connector, gateway, or Information Store schema just by using their identifiers. i2 Analyze searches for types in all three schemas, in that order. It is also possible to specify which schema a particular item type comes from:

- When you are specifying the default return types for the connector as a whole, you can set the `entityTypeLocation` and `linkTypeLocation` properties.
- When you are specifying the possible return types from a service, you can provide locations in the `resultItemTypeIds` structure.
- In each record that you return from a service, you can set the `typeLocation` property.

When you deploy a connector, you can use the `<connector>` element in the topology file to override some of the schema settings in the connector. You can instruct the connector to use a gateway schema with a different short name, which can be useful if you were unable to deploy the schema with the name that the connector expected. You can also change the short name of the connector schema, which is useful if you have two connectors that advertise the same name.

5. To specify a gateway schema with a different short name, add the `gateway-schema` attribute to the `<connector>` element:

```

<connectors>
  <connector id="ConnectorId"
            name="ConnectorName"
            base-url="Protocol://HostName:PortNumber"
            gateway-schema="ShortName"/>
</connectors>

```

6. To provide a different short name for the connector schema, add the `schema-short-name` attribute to the `<connector>` element:

```
<connectors>
  <connector id="ConnectorId"
    name="ConnectorName"
    base-url="Protocol://HostName:PortNumber"
    schema-short-name="NewShortName" />
</connectors>
```

After you make changes to the topology of an i2 Analyze deployment, you must update the deployed connectors configuration and restart the i2 Analyze server, as described in [“Adding a connector to the topology”](#) on page 316.

Converting records between types

A deployment of i2 Analyze that defines a large number of item types across several schemas can become confusing for administrators and users, and might even compromise the utility of the platform. To reduce the number of types in the system, and to enable richer analysis, i2 Analyze supports the dynamic conversion of records from one type to another.

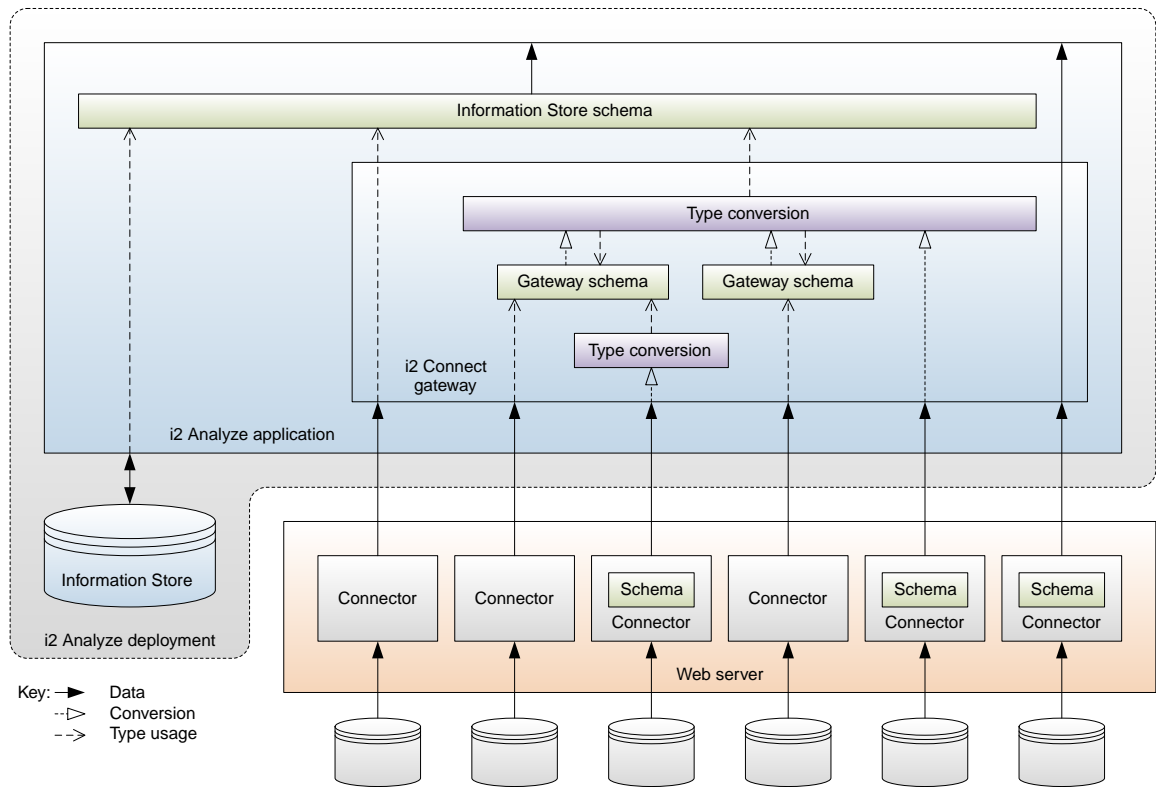
In a deployment of i2 Analyze that includes several schemas, and especially one that features connector and gateway schemas from third parties, there can be multiple item types with similar names or purposes. For example, two connector or gateway schemas might independently define similar "Person" item types, which might in turn be slightly different from a "Person" type defined in the Information Store schema.

This kind of arrangement can have a number of consequences:

- The existence of several item types with the same or similar names can be confusing for administrators and for users, who see records on their charts and item types in their palettes that look the same, but are not.
- By the same token, the analytical tools in i2 Analyze and IBM i2 Analyst's Notebook Premium do not consider records with similar-looking types to be equivalent for the purposes of comparison and other functions.
- Only records that have types from the Information Store schema can be uploaded to the Information Store.

To address these problems, the i2 Connect gateway provides a dynamic type conversion service. At startup, the gateway builds a catalog of all the item types in all the schemas in the deployment. You can then create mappings from one item type to another, in the following combinations:

- Connector schema type to Information Store schema type
- Connector schema type to gateway schema type
- Gateway schema type to Information Store schema type
- Gateway schema type to gateway schema type (in a different gateway schema)



Type conversion mappings describe how the property values of source records that have a particular item type become the property values of target records that have a different item type.

For example, a gateway schema might define an entity type named "Person", with property types including "Forename" and "Surname". In the same deployment, the Information Store might also define an entity type named "Person", with property types including "First Name" and "Family Name". Provided that the property types in question have compatible logical types, you can create a mapping to change the type of any record with the gateway type so that it becomes a record with the Information Store type.

The effect of this type conversion mapping is that any record that would otherwise appear in search results or on the chart surface with the gateway type instead appears with the Information Store type. As a result, it can be compared directly with existing records in the Information Store, and can itself be uploaded to the Information Store if the user requires it.

Note: It is not mandatory to create mappings between item types. By default, records from connectors are copied to users' charts with their assigned entity or link types, without modification.

To create type conversion mappings, you use the i2 Analyze Server Admin Console.

1. In a web browser, navigate to the URI of a i2 Analyze development environment, but append / admin.
For example: `http://host_name:9082/opa1/admin`.

When you log in as a user with administrator privileges, the i2 Analyze Server Admin Console appears.

2. On the left of the user interface, select the **i2 Analyze Type Conversion** app.

In the main panel, the app displays all the item types from all the connector and gateway schemas in this deployment of i2 Analyze. (Item types from an Information Store schema cannot be source types in type conversion mappings.) For each type in the list, you can create a mapping to a different type.

3. Select a source item type, and click **Create mapping** to display the **Convert records** dialog.

This dialog displays all the item types from all the gateway schemas and any Information Store schema in this deployment of i2 Analyze. (Item types from connector schemas cannot be target types in type conversion mappings.)

4. Select a target type for your mapping, and click **Create mapping** to display the dialog with the same name.

The **Create mapping** dialog displays a list of the property types of the *target* item type. For each property type, you can elect to provide no mapping, or to say that a property with this type is to be set from a source property with a specified type, or that a property with this type is always to be set to a fixed value.

5. Configure the property type mappings as you see fit, and then click **OK** to create the item type mapping.

If there are problems with any of the mappings that you create, the app displays inline warning and error messages.

As you develop your type conversion mappings, you can preview their effects on the results that users see when they query external data sources by temporarily applying them to the development server.

6. Click **Apply** to apply all the current type conversion mappings to the server.
7. Click **Preview services** to display the same **Services** dialog that users see when they query external data sources.

Note: If any of your connectors support user-specific configuration, you might need to log in with different credentials in order to see the effect of your mappings on all services.

8. Select and run some of the queries.

The descriptions and results of the queries now reflect the mapping that you created. Any references to the source types in your mappings are replaced with their target types.

This approach to previewing the effect of your mappings is not suitable for a production environment, where you must instead deploy a type conversion mapping configuration file to your i2 Analyze servers.

9. When you are satisfied with the mappings, click **Export** to create a configuration file named `mapping-configuration.json`.
10. Save the configuration file to the `configuration\fragments\common\WEB-INF\classes` folder of the i2 Analyze deployment toolkit.
11. Redeploy i2 Analyze to update the application with this change to its configuration. In a command prompt, navigate to the `toolkit\scripts` directory and run the following commands:

```
setup -t stopLiberty
setup -t deployLiberty
setup -t startLiberty
```

12. Finally, verify that the changes that you made in the development environment behave correctly in the production environment.

Supporting seeded queries

If you configure a connector's services to enable it, users can run queries with *seed records* that enable or change their functionality. Your implementations of the acquire endpoint receive identity and property data from selected records that you can use to drive operations against the external data source.

As you read the following information, also look at the `config.json` files in the example projects at <https://github.com/ibm-i2/analyze-connect>, some of which demonstrate the results of following this procedure.

The behavior of a query that supports parameters is modified by values that users specify when they run it. The behavior of a query that supports seeds is modified by the records that a user selects. Seeded queries generally fall into one of a handful of categories:

- A "find like this" query looks at the property values of a selected record and searches for data in the external source that has the same or similar property values.
- A "get latest" query uses the identifying information of a selected record to look for the same data in the external source, with the aim of updating the record.
- An "expand" query uses the identifying information of a selected record to search for data in the external source that is connected to that record.
- A "find path" query receives the identifying information for exactly two selected records and searches for a path that connects those records in the external source.

To indicate that a particular service supports seeds, you add a `seedConstraints` object to its definition in your response from the configuration (or user configuration) endpoint. An empty object indicates that the service supports any number of *entity* seeds. To change this behavior, and for more control over the constraints, you can populate the object. The REST SPI documentation for the configuration endpoint describes its structure:

```
"seedConstraints": {
  "connectorIds": [""],
  "min": 0,
  "max": 0,
  "seedTypes": {
    "allowedTypes": "",
    "itemTypes": [
      {
        "id": "",
        "min": 0,
        "max": 0,
        "typeLocation": ""
      }
    ]
  }
}
```

A service can specify which records it accepts as seeds by restricting the connectors they came from. It can also set boundaries on the number of seed records it accepts, regardless of their type. A service can specify whether entity or link records are allowed as seeds (entity records are the default), and then restrict the range to a subset of that group. It can also set boundaries on the number of seeds, on a per-type basis.

If you configure the constraints so that requests must contain at least one seed of each permitted item type, then the service *requires* seeds. Users cannot run the service without an appropriate

selection. Otherwise, your implementation of the acquire endpoint must support requests that might or might not contain seed information.

When the i2 Connect gateway calls your acquire endpoint to perform a seeded query, it includes a payload that contains identifiers and property values for all seeds. For information about the structure of the payload, see the REST SPI documentation for the request parameter of the acquire endpoint.

In outline, the procedure for supporting seeded queries in one of your services is to start by adding seed constraints to its configuration. Clients then interpret the configuration and present the queries to users. When users run a query, your implementation of the acquire endpoint receives seeds that you can use to guide your response.

1. Decide what kind of seeded operation you want to perform, and its implications for the service.

For an "expand" query, for example, you might accept a fairly large number of entities of any type as seeds. For "find like this", the seed is more likely to be a single record of a specific type.

2. Add a `seedConstraints` object that reflects the requirements of the query, to the response from the configuration (or user configuration) endpoint.
3. Implement the acquire endpoint for your service.

If at least one of the constraints does not specify a minimum record count, the endpoint might be called with or without seeds. The REST SPI documentation for the acquire endpoint describes the structure of the seed data.

Important: For "get latest" and "expand" queries, it is common for seed records also to appear in the results. In that case, you must ensure that the `id` of the outgoing record matches the `seedId` of the incoming record. In an "expand" query, for example, you can identify a seed as being one end of a link in the response by setting the `fromEndId` of the link to the `seedId`.

4. Restart the i2 Analyze server or instruct the i2 Connect gateway to reload the configuration. Then, connect to the server with a client, log in as a user who can see the query, and ensure that the service is working properly.

Supporting parameters

In nearly all real situations, users want to be able to customize the queries that they can run against an external data source. To make that possible, you must configure the service to present parameters to its users. In your implementation of the acquire endpoint, you can act on the values that they specify.

As you read the following information, also look at the `config.json` files in the example projects at <https://github.com/IBM-i2/Analyze-Connect>, some of which demonstrate the results of following this procedure.

To support parameters in a service, you add a set of conditions for which users can supply values. In a connector that supports user-specific configuration, you can even arrange for the same service to support different parameters, depending on who is using it.

For example, in a query for people, you might allow users to search for particular names or characteristics. In a "find path" seeded query, you might allow users to specify how long the path can be - and you might vary the limit according to the user's group membership.

Note: A significant difference between visual query and external searches is that in the latter, conditions are not bound to property types. A condition in a service can be anything that you can process usefully in your implementation.

To add parameters to a service, you add a *client configuration* to its definition that describes how each condition is presented to users. The REST SPI for the configuration endpoint describes the structure of `clientConfig` objects, which look like this outline:

```
{
  "id": "",
  "config": {
    "sections": [
      {
        "title": "",
        "conditions": [
          {
            "id": "",
            "label": "",
            "description": "",
            "mandatory": false,
            "logicalType": ""
          }
        ]
      }
    ]
  }
}
```

The client configuration is responsible for the appearance of the conditions that users see, and the restrictions on the values they can provide. As well as controlling the types of conditions, and specifying whether supplying a value is mandatory, you can add further validation that is appropriate to their type. For more information about this kind of validation, see the REST SPI documentation for the configuration endpoint.

When a user opens a query that supports parameters, they see a form that displays your conditions. When they provide values and run the query, your implementation of the acquire endpoint receives a payload that contains those values. You can write the implementation to act on those values in any way that makes sense for your data source.

Like many other aspects of developing a connector, supporting parameters means changing your implementations of the configuration (or user configuration) and acquire endpoints.

1. In your response from the configuration endpoint, write or modify the service definition so that its `clientConfigType` is "FORM", and add a `clientConfigId`.

This identifier links to a client configuration elsewhere in the response. In some circumstances, it might be appropriate to use the same client configuration for more than one service.

2. Add a `clientConfigs` array to the response, and within it a client configuration object whose `id` matches the identifier that you specified in Step 1.
3. Add your conditions to the client configuration.

To begin with, you might consider leaving out validation checks in the interests of getting a working implementation quickly.

4. Add code to your implementation of the acquire endpoint that unpacks the condition values from the payload.

You can then use those values to affect the query that you perform against the external source.

5. Restart the i2 Analyze server or instruct the i2 Connect gateway to reload the configuration.

You can now test the code and make sure that it does what you expect.

6. Return to the response from the configuration endpoint, edit the condition descriptions, and add client-side validation to improve the user experience.
7. Restart or reload again, and ensure that the validation has your intended effect.

The validation that you can specify in the response from the configuration endpoint is performed by the client, and applies to values in isolation. If your service (and your users) might benefit from some more complex validation, consider adding a validate endpoint.

Supporting validation

The support in the i2 Connect gateway for queries that take parameters includes the ability to perform relatively simple, client-side validation on the values that users supply. In cases where running a query might fail due to a set of values that are mutually incompatible, you can write an endpoint to perform server-side validation.

When you write a service definition that includes conditions, you have the option (and sometimes the duty) to include logic that validates the values that users supply for those conditions. For example, you might insist that a string is shorter than a certain length, or that its contents match a particular pattern. If you ask the user to select from a discrete set of values, then you must provide the values for them to select from.

However, there are other kinds of validation that the mechanism for defining conditions does not support. In particular, you cannot use it to validate values that are reasonable in isolation, but faulty in combination. (For example, dates of birth and death might both contain reasonable values, but it would make no sense to search for individuals where the latter is earlier than the former.)

If you have an implementation of the acquire endpoint for which it might be useful to perform this kind of validation, you can write a validate endpoint. When your configuration specifies a validate endpoint for a service, the gateway uses it before the acquire endpoint, and passes the same payload. If you decide that validation fails, the request to the acquire endpoint does not happen, and the user sees an error message that you specify.

To add a validate endpoint to a service:

1. In your response from the configuration (or user configuration) endpoint, add the `validateUrl` setting alongside the existing `acquireUrl` setting. Set its value to the location of the implementation.
2. Implement the rules for the validate endpoint in a way that is consistent with its definition in the REST SPI documentation.
The payload that the endpoint receives in the request contains all the same seed and parameter information as the acquire endpoint receives.
3. If validation succeeds according to your rules, return an empty response. If it fails, set the response to a simple object that contains an `errorMessage`:

```
{
  "errorMessage": ""
}
```

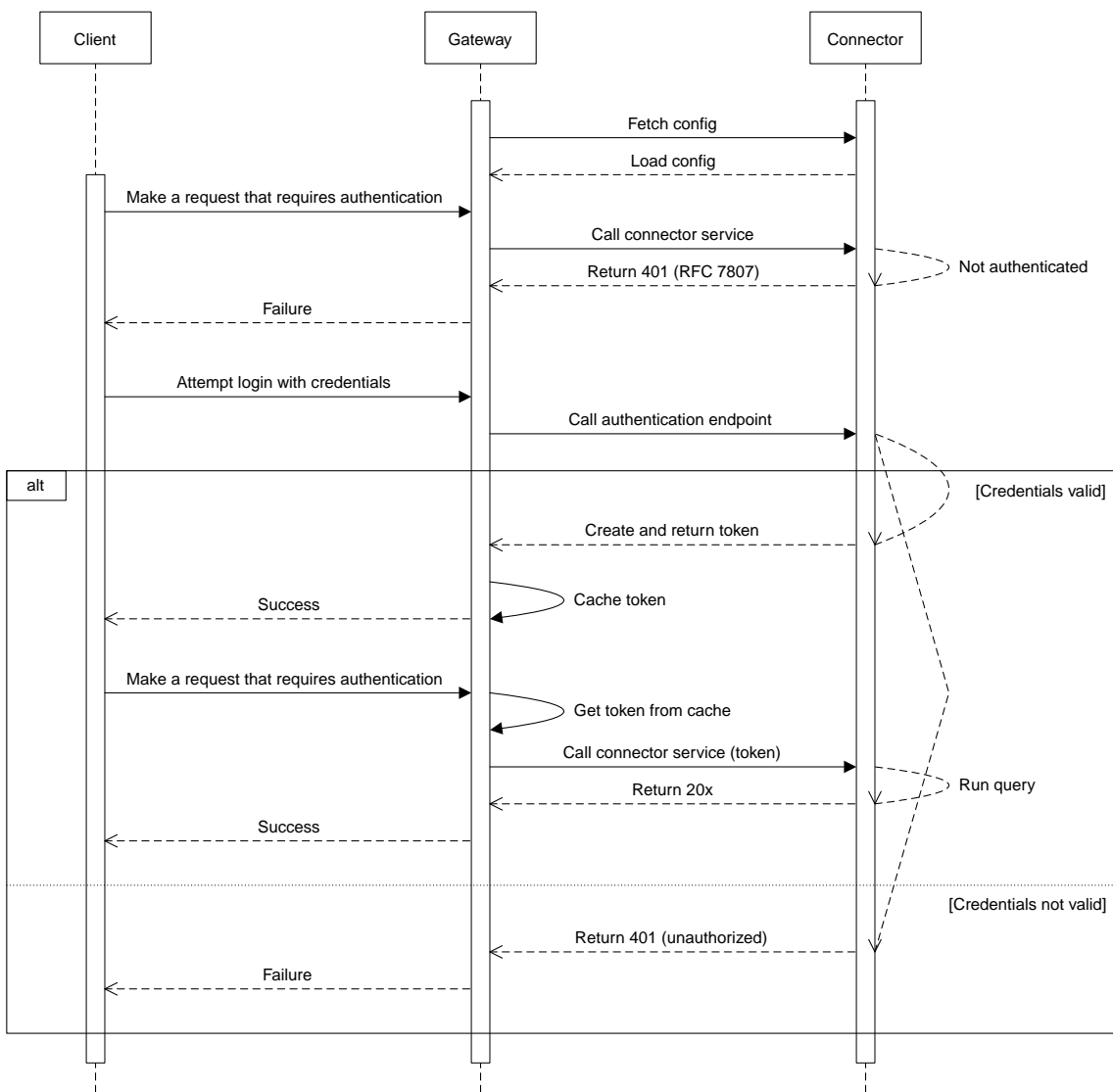
When the i2 Connect gateway receives a response that is not empty, it does not then send a request to the acquire endpoint.

4. Restart the i2 Analyze server or instruct the i2 Connect gateway to reload the configuration. Test that your new validation has the correct behavior.

Supporting authentication

Many of the data sources that users might want to query through a connector require authentication. For example, some sources require a username–password combination, while others need the caller to provide an API key. The i2 Connect gateway supports asking users for credentials on behalf of an i2 Connect service, and management of authenticated connections during a user session.

To enable authentication, a connector must specify in its configuration which of its services require authentication. If a user makes a request for an authenticated service without previously providing credentials, the gateway's response prompts the client to display a login dialog. When the user provides valid credentials, the gateway caches a token in memory that allows further requests to succeed.



Configuring a service to require authentication is similar to configuring it to take parameters. In the response from the connector's configuration endpoint, you define a form that prompts the user for credentials. To support the interactions that the i2 Connect gateway expects, you must implement a login method on the connector that validates the user's credentials. You must also implement the token-handling behavior on any affected service endpoint.

Note: Although both can function as ways of controlling access to i2 Connect services, supporting authentication is orthogonal to supporting user-specific configuration. Typically, user-specific configuration affects the availability (or functionality) of a service according to rules set by the i2 Analyze server, while authentication controls users' ability to use a service according to rules set by the service itself.

For a more detailed description and a demonstration of setting up authentication for an i2 Connect service, see the examples and documentation at <https://github.com/ibm-i2/analyze-connect>. The following steps provide an outline of the procedure.

1. In your response from the configuration (or user configuration) endpoint, write the service definition so that it contains an `authConfigId`.

This identifier links to an authentication configuration elsewhere in the response. In some circumstances, you might use the same authentication configuration for more than one service.

2. Add an `authConfigs` array to the response, and within it an authentication configuration object whose `id` matches the identifier that you specified in Step 1.

This object contains all the information that the i2 Connect gateway needs to get credentials from the user and call an authentication endpoint on the connector.

For example:

```
{
  "id": "authConfig1",
  "loginUrl": "/login/userpass",
  "form": {
    "description": "This service requires a username and a password.",
    "fields": [
      {
        "id": "username",
        "label": "Username",
        "type": "text"
      },
      {
        "id": "password",
        "label": "Password",
        "type": "password"
      }
    ]
  }
}
```

3. Create the endpoint at the `loginUrl` that is defined in the authentication configuration.

The syntax of this POST method is straightforward: it receives the credentials that the user provides, and (on success) responds with a token for subsequent requests to use. The detail of generating the token depends on the nature of the service. For example, you might be passing the credentials to an external provider, or implementing the authentication yourself in the service code.

If authentication fails, the endpoint must instead respond with an object that complies with [RFC 7807](#). For an example of doing so, see the file named `ExternalConnectorDataService.java` in the `connector/auth` sample project at <https://github.com/ibm-i2/analyze-connect>.

4. Adapt any endpoint that requires authentication (for example, the `acquire`, `delete`, or `results` endpoints) to use the token mechanism.

The implementations should expect requests to include an authorization header containing a token that was originally returned by the authentication endpoint, in the form `Authorization: Bearer token`.

If the token is valid, processing of the request should proceed normally. If the token is not valid, the endpoint must respond in the same way as the authentication endpoint. The user is then required to re-authenticate with the service.

5. Restart the `i2 Analyze` server or instruct the `i2 Connect` gateway to reload the configuration. Connect to the server with a client that supports external searches, and verify that the connector and its services behave in the way you expect.

Source identifiers

In all versions of `i2 Analyze`, the ETL pipeline requires you to provide origin identifiers for incoming records through the ingestion mappings that you write. Starting from `i2 Analyze 4.3.5`, developers of connectors for `i2 Connect` and plug-ins for `i2 Analyst's Notebook Premium` can explicitly provide source identifiers for the records that they create.

Note: For high-level descriptions of source identifiers and origin identifiers (and the differences between them), see [Identifiers in `i2 Analyze` records](#).

When you attach recognizable identifiers to the records that you create in connector code, you enable services to perform operations based on those identifiers. You also enable `i2 Analyze` clients and servers to perform matching on records that have a shared source.

The rules that govern the structure and contents of source identifiers are similar to - but not the same as - the rules for [origin identifiers](#). In line with their definitions, the rules for source identifiers are sometimes less restrictive than those for origin identifiers.

The structure of a source identifier

A source identifier contains a *type* and a *key*. If a source identifier is attached to a record that is subsequently uploaded to the Information Store, then the identifier is stored with that record in the database. As a result, there are limits on how much information types and keys can store.

type

The *type* of a source identifier allows the services in an `i2 Analyze` deployment to determine whether the source identifier is "known" to them - that is, that they can understand the key.

The value of the type element does not have to be meaningful, but should be unique to your services so that you avoid clashes with any third party services you might use.

- The length of the source identifier's type must not exceed 200 bytes, which is equivalent to 100 2-byte Unicode characters.
- The following types are reserved and must not be used (case-insensitive):
 - `OI.IS`
 - `OI.DAOD`
 - `OI.ANB`
 - Anything starting with `i2`.

key

The *key* of a source identifier is an array containing the information necessary to reference the data in its source. The pieces of information that you use to make up the key differ depending on the source of the data.

- The total length of the source identifier's key must not exceed 692 bytes, which is equivalent to 346 2-byte Unicode characters.
- The key is stored as a serialized JSON array, so additional characters appear alongside the actual key elements: two characters are required for the array braces, and two quotes are required for each element, with commas as separators between elements.

In other words, a key with N elements requires $3N + 1$ characters of overhead. For example, the total length of ["a", "bc", "defg"] is 17 characters, while ["a, bc, defg"] is 13 characters. Also, if some special characters are present in a key element, they are escaped for storage. For example, " becomes \", which further increases the size of the key.

Using source identifiers

When a service receives a seed record, it can inspect the source identifiers to find out whether a known type is present. If it is, then the key can be used to retrieve or match information from the source system.

When records are returned from external sources, the client can match them against other records by using the source identifiers (if source identifier matching is enabled in the match rules with `enableSourceIdentifierMatching="true"`). Also, if source identifier matching is configured in the system match rules, then source identifiers provided through i2 Connect or Analyst's Notebook Premium can be matched against the origin identifiers of records in the Information Store.

In both cases, source identifier matches occur when record item types match *and* the type and keys of the identifiers are an exact (case-sensitive) match.

Note: After upgrading to i2 Analyze 4.3.5 from a previous release, if you want to match against the origin identifiers of already ingested records, you must build a new match index.

Limitations

There is a limit on the number of unique source identifiers that you can add to a record in the Information Store. The default limit is 50, but you can modify it by adding `MaxSourceIdentifiersPerRecord=N` (where N is a positive integer) to the `DiscoServerSettingsCommon.properties` file.

Origin identifiers are not stored in chart records. As a result, they are not present as source identifiers on seeds that are sent to connectors or Analyst's Notebook Premium plug-ins. Only those source identifiers that have been added to records through the same mechanisms are present.

Connectors that specify source identifiers are API-compatible with older versions of the i2 Connect gateway and Analyst's Notebook Premium, but those products will not recognize new source identifiers as such. Rather, they will be treated as ordinary identifiers or ignored. To use source identifiers to their full potential, upgrade all products to the latest release at your earliest opportunity.

Valid formats for i2 Connect data values

This document describes how to format the i2 Connect data that you return from a connector or use in default values in the connector configuration. An example is given for each logical type.

SINGLE_LINE_STRING

Single-line strings can be up to 250 bytes long, and can be further limited by the schema.

```
"This is a single-line string."
```

MULTIPLE_LINE_STRING

Multiple-line strings can be up to 32K bytes long, and can be further limited by the schema.

```
"This is a multiple-line string."
```

DATE

Date values must be in the ISO 6801 format "YYYY-MM-DD". Also, values must be in the range from 1753-01-01 to 9999-12-30.

```
"2021-11-30"
```

TIME

Time values must be in the ISO 6801 format "hh:mm:ss".

```
"23:59:59"
```

DATE_AND_TIME

There are two ways to return values with the DATE_AND_TIME logical type. You can use an ISO 8601 string *without* a time zone, or a date-and-time JSON object that explicitly defines the time zone.

When you use the ISO 8601 format, the time zone is taken from the `timeZoneId` that's defined in `defaultValues` in the connector configuration. If no default is present, the time zone is set to UTC.

```
"2021-12-30T23:59:59"
```

or

```
"2021-12-30T23:59:59.999"
```

or

```
{
  "localDateAndTime": "2021-12-30T23:59:59.999",
  "isDST": false,
  "timeZoneId": "Europe/London"
}
```

BOOLEAN

Boolean values must be either `true` or `false`.

```
true
```

or

```
false
```

INTEGER

Integer values must be in the range from -2147483648 to 2147483647.

```
2147483647
```

DOUBLE

Double values must be in the range from 4.94065645841246544e-324d to 1.79769313486231570e+308d.

```
4.94065645841246544e-324d
```

DECIMAL

Decimal values can contain up to 18 digits before the decimal separator, and up to 4 digits after it. There can be a leading minus sign, but no exponent (e) notation.

```
-123456789012345678.1234
```

SELECTED_FROM

Selected-from string values must match a permitted value that the schema or a form condition defines.

```
"This is a selected-from string."
```

SUGGESTED_FROM

Suggested-from string values must match a permitted value that a form condition defines when you use them as default values, but are otherwise unrestricted.

```
"This is a suggested-from string."
```

GEOSPATIAL

Geospatial values must be formatted as GeoJSON points, as described at <https://datatracker.ietf.org/doc/html/rfc7946#section-3.1.2>.

The first element in the coordinates array is longitude, and must be a decimal between -180.0 and 180.0. The second element is latitude and must be a decimal between -90.0 and 90.0.

```
{
  "type": "Point",
  "coordinates": [1.0, 2.0]
}
```

GEOSPATIAL_AREA

Geospatial area values must be formatted as GeoJSON feature collections, as described at <https://datatracker.ietf.org/doc/html/rfc7946#section-3.3>. Foreign members are allowed.

The geometry must be of type Polygon or MultiPolygon, both of which must contain at least four coordinates, where the first coordinate matches the last.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [1.0, 2.0],
              [3.0, 4.0],
              [5.0, 6.0],
              [1.0, 2.0]
            ]
          ]
        ]
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [1.0, 2.0],
            [3.0, 4.0],
            [5.0, 6.0],
            [1.0, 2.0]
          ]
        ]
      }
    }
  ]
}
```

IBM i2 Analyze Schema Designer

Welcome to the IBM i2 Analyze Schema Designer documentation, where you can find information about how to use IBM i2 Analyze Schema Designer.

Getting started

[“Schemas” on page 341](#)

A schema defines the structure of data that will be stored within your system. A schema must be designed in a way that effectively maps the data that you have, to the type of analysis that is going to be carried out.

[“Charting schemes” on page 350](#)

Charting schemes determine the mapping between items in i2 Analyze and the same items when displayed on a chart in Analyst's Notebook Premium. You can transform data in different ways by creating multiple charting schemes.

Common tasks

[“Creating entity types” on page 344](#)

Entity types define the structure for data that will be stored in i2 Analyze. By creating entity types, you can classify your data.

[“Creating link types” on page 347](#)

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

[“Changing the grading scheme” on page 356](#)

You can grade your properties in i2 Analyze to indicate the level of confidence you have in property values. You can create and configure grade types within a grading scheme.

Troubleshooting and support

[IBM Software Support home page](#)

More information

[IBM training](#)

Notices and contact information

Copyright

© Copyright IBM Corporation 2012, 2021.

All rights reserved. May only be used pursuant to an IBM software license agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of IBM Corporation. IBM Corporation grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the IBM Corporation copyright notice. No other rights under copyright are granted without prior written permission of IBM Corporation. The document is not intended for production and is furnished “as is” without warranty of any kind.

All warranties on this document are hereby disclaimed, including the warranty of non-infringement and the implied warranties of merchantability and fitness for a particular purpose.

Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM United Kingdom Limited
Hursley House
Hursley Park
Winchester, Hants, SO21 2JN
UK*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users to resolve problems with the product.

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM software subscription and support contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

To contact IBM Support about a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM Support topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
 - Online through the IBM Support Portal at <https://www.ibm.com/mysupport/>. You can open, update, and view all of your service requests from the Service Request portlet on the Service Request page.
 - By phone. For the phone number to call in your region, see the Directory of worldwide contacts web page at <https://www.ibm.com/planetwide/>

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. To submit any comments about this book or any other documentation, send your comments by email to ibmi2doc@uk.ibm.com. Be sure to include the name of the book, the part number of the book, the version of the software, and, if applicable, the specific location of the text that you are commenting on (for example, a page number, table number, or a heading).

Schemas

A schema defines the structure of data that will be stored within your system. A schema must be designed in a way that effectively maps the data that you have, to the type of analysis that is going to be carried out.

Structuring your data in a way that is easy to understand is important because correctly organizing data increases the likelihood of finding key information that relates to your investigation.

Data that will be stored within your system must be defined by using an entity, link, and property-based model. Structuring your data in this format to fit the specifications of your investigation is called data modeling.

You can use the Schema Designer to classify your data to resemble your data model. Data classification involves creating components within your schema for the data that will be stored in terms of entity and link types, with their property types.

Creating an effective schema involves several cycles of reviewing and testing before it is uploaded into a production environment. This process reduces the need for changes during production that can affect the data within your system.

Related concepts

Charting schemes

Charting schemes determine the mapping between items in i2 Analyze and the same items when displayed on a chart in Analyst's Notebook Premium. You can transform data in different ways by creating multiple charting schemes.

Related tasks

Creating schemas

When the data that you want to analyze is modeled in terms of entities, links, and properties, you can create a schema with schema components. The example schemas that come with i2 Analyze can be edited if you do not want to start from a blank schema.

Data modeling

Before you create your schema, you must analyze the data that is likely to be available for analysis and understand how that data is used during an investigation. Based on your findings, you can create a data model that defines your data in terms of entities, links, and properties.

It is important to first establish the aim of your investigation when you are creating a data model because it can help organize data effectively. You can analyze your data and consider how it fits the basic concepts of data modeling.

The basic concepts of data modeling are formed by entities, links, and properties (ELP). Real world objects in your data can be expressed as entities, the association between those entities as links, and the values that characterize the entities and links as properties. Both entities and links are types of items; items contain properties and property groups, and properties can be placed in items or property groups.

The Schema Designer can be used to create a schema that is based on the data model's specifications.

Important: If your deployment is using data connectors with other sources of data, the shape of the data from those data connectors must also be modeled in your schema.

Schema structure

You can determine the structure of your schema by analyzing your data in terms of item and property types. The schema structure is based on your data model.

Item types form the core of your schema and are created to classify the entities and links your identified in your data model. The following table describes the item types that you can create:

<i>Table 2. Item types</i>	
Name	Description
Entity type	You can create entity types to classify your entities that represent your real world objects. For example, you might create a Person entity type to classify entities that represent specific people in an investigation. When you are adding information about these people, details about a specific person are added to a separate entity of type person within the system. You can assign a semantic type to an entity type to provide it with meaning by identifying the nature of the data.
Link type	You can create link types to classify the links you create between your entities. For example, a Car entity type and a Person entity type might be associated with an Owner link type. When you are adding information about the relationship between a car and a person, it is added as a link of type owner within the system. A link type also dictates what types of entities can be at the ends of links of that type. You can assign a semantic type to a link type to provide it with meaning by identifying the nature of the data.

When you create item types, you can also create property and property group types that define information about that item type. The following table describes how property and property group types are used to classify property values for that are found in your data:

<i>Table 3. Property and property group types</i>	
Name	Description
Property type	You can create property types to identify features in entity and link types that might be useful in analysis. Property types are important because they enable comparison of data more effectively by declaring whether they are the same. For example, you can create an Eye Color property type to classify properties in your data that represents different eye colors. The logical type of a property type establishes what type of data can be stored in that property type.
Property group type	Property types can be grouped in property group types to organize your data in a way that might help analysis. For example, if you had a Person entity type, you might assign facial features such as Hair Color or Eye Color as property types, and these types could be grouped in a Facial Features property group type.

In addition to creating item, property, and property group types, the following table describes how your schema can define other behavior:

<i>Table 4. Other behavior</i>	
Name	Description
Labeling scheme	Labeling schemes define the property types that are used to identify items.

<i>Table 4. Other behavior (continued)</i>	
Name	Description
Grade types	You can add grading information to indicate the level of confidence you have in your property values for items. You can create a grading system with appropriate grade types to match the rating system that your organization uses.
Link strengths	Link strengths that are set in the schema can be used on all link types. Only one link strength can be set as default. The default link strengths describe the links as being either confirmed, tentative, or unconfirmed, but you can configure the labels and their appearance in the Schema Designer.

Related tasks

Assigning semantic types

You can assign entity, link, and property types a semantic type to identify the real-world meaning of the data. Ensuring that your semantic types are assigned accurately is important because it affects the organization and interpretation of your data.

Schema development process

When you are creating a schema, you can go through a number of iterations to ensure that the schema satisfies the requirements of your organization and functions correctly. By developing your schema in test environments, you ensure that it fits the format of your investigation.

Ensuring that you create and include only components that are necessary to your investigation is important. You cannot delete unnecessary components from your schema after you deploy it to the Information Store.

After your schema is configured to the specifications of your data model, you can create one or more charting schemes. A charting scheme determines how items that are stored in i2 Analyze are mapped to your chart.

Schema validation

To ensure the schema functions and is safe to use in your production environment, there are restrictions your schema must comply with when you create and develop it. The number and types of restrictions that are imposed on your schema vary depending on what stage of the development it is in.

Validation checks prevent you from implementing any changes that involve removing a unique ID from your schema. Your schema must retain these unique IDs because without them you cannot find the data that they are assigned to.

Your schema must pass three types of validation checks before it can be uploaded:

System type validation

When you create a new schema, it contains system entity types. The system entity types must exist for the schema to function. You can configure the Name and Description sections of these system entity types as they do not change the function of the components. You can assign other components to the system entity types such as property and property group types. You are unable to remove them from the schema.

Validation as you edit your schema

While you edit your schema, it must pass the Schema Designer validation checks. These checks do not prevent you from editing your schema, but make you aware of any invalid changes that are made. These validation checks ensure that your schema contains all the necessary components

for it to function, such as link end constraints. Any issues that are found in the schema can be found at the bottom of the **Schema Designer** window for you to fix.

Validation when you upload your schema

To protect your data, i2 Analyze checks ensure that any changes you might make do not result in data loss. Any changes that can cause data loss are called destructive changes. If you try to upload a destructively changed schema, the validation checks fail, and an error message is displayed.

Related tasks

Creating link types

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

Changing link strengths

You can create link strengths to visually indicate how much confidence there is in the information that is represented by a particular link. Link strengths that you create in the schema can be applied to all link types.

Creating schemas

When the data that you want to analyze is modeled in terms of entities, links, and properties, you can create a schema with schema components. The example schemas that come with i2 Analyze can be edited if you do not want to start from a blank schema.

When you create a schema, it contains default system entity types. These components hold necessary information for the system and exist in order for it to work.

You can create schema components to classify and structure your data when it is stored in i2 Analyze. You can classify your data in terms of item types with their property types.

Related concepts

Schemas

A schema defines the structure of data that will be stored within your system. A schema must be designed in a way that effectively maps the data that you have, to the type of analysis that is going to be carried out.

Creating entity types

Entity types define the structure for data that will be stored in i2 Analyze. By creating entity types, you can classify your data.

To create an entity type:

1. In the **Schema Designer** window, click **Insert > New Entity Type**.

A new entity type is added to the navigation tree. The new entity type's configurable fields are displayed in the Entity type tab. The entity type is assigned a default name and an icon.

2. Enter a name and description, and select an icon for your new entity type.
3. Assign a suitable semantic type that sufficiently defines the entity type.
4. Create property types for the entity type.

Your entity type is successfully configured, with your property and semantic type specifications.

Creating property types

Property types allow you to classify the data that relates to a particular item type. Create a property type for each aspect of the data that you would like to be able to analyze.

To create a property type:

1. To insert a property type into an item type, in the **Schema Designer** window, click **Insert > New Property Type**. You now have a property type. To configure the property type, complete the following fields. Repeat this step for each property type that you add to your item type:

<note>Property types can be added both within existing property group types, or as stand-alone property types.</note>

Field	Description
Name	The name is used to identify your property type in the schema and Schema Designer window. It has no size or content limitations, however you might want to keep it short as it can be used as a column header in results grids.
Description	This field is used to store the hover help associated with the property type.
Semantic Type	Indicates the semantic type of the property.
Logical Type	The type of data that is stored in the property type and must be defined.
Is Mandatory	Select this option to ensure that an item includes at least one value for this property.

2. Optional: Click **Is Mandatory** if you want to make the property type mandatory.

Your property type is created. The property type can be integrated into a labeling scheme, graded, and also mapped to chart item property types in a charting scheme.

Related tasks

Creating link types

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

Creating property group types

Property groups are used to group related properties within the schema. You can create property group types to group the property types that you consider to be similar.

When you group related properties as part of your schema design, it makes them easier to view and manage for schema maintainers. Property groups are not available in Information Store or i2 Connect schemas.

To create a property group type:

1. To add a property group type to your selected item type, click **Insert > New Property Group Type**. You now have a property group type. To configure the property group type, complete the following fields. Repeat this step for each property group type that you add to an item:

Field	Description
Name	The name is used to identify your property type in the schema and Schema Designer window. It has no size or content limitations, however you might want to keep it short as it can be used as a column header in results grids.
Description	This field is used to store a description of the property group type that might be useful to schema maintainers.
Is Mandatory	Select this option to ensure that at least one property within the property group is populated for an item.

2. Optional: Click **Is Mandatory** if you want to make the property group type mandatory.

Your property group type is created.

Specifying logical types

You must specify the logical type of a property type to inform your deployment about the type of data it is processing so it can enable and disable operators. By specifying the logical type, you can enter or import data more accurately through validation checking.

The type of data that is stored in a property is determined by the logical type. It is important to select the correct logical type for the type of data that you would like to store and the type of analysis that is needed. For more information about the logical types that you can choose, see [“Logical types” on page 346](#).

Note: When you are deciding whether to use a single-line or multi-line string type, it is important to consider the intended use of the data and the performance of your system.

A property with the single-line string logical type is used to store text that can be used for faceting, and supports operations such as 'Starts with' and 'Ends with'. A property with the multi-line string logical type can be used to store larger blocks of information, but the values cannot be used for faceting or operations such as 'Start with' and 'Ends with'.

A property with the single-line string logical type can contain a maximum of 8000 characters in Db2 databases, or 4000 characters in SQL Server databases.

To specify a logical type:

1. In the tab of your chosen property type, select a logical type from the **Logical type** list. The default logical type is **Single Line String**.
2. If you select either **Selected From List** or **Suggested From List**, add entries to the Selection List. You can input each entry manually by clicking Add, or from an existing list file by clicking Import.

Note: When you import entries from a list file:

- They must be present in a text document (.txt).
- They must be presented as a list with the following structure:

entry name	tab	description
entry name	tab	description

- The first line of the import file is ignored.

3. Optional: For the **Single Line String**, you can specify the **Maximum Length**.

Note: To help reduce performance issues, the default length for single-line strings is 250 characters, increasing this limit affects performance and must be tested.

Logical types

The logical types that you can specify in your schema depend on the data store that your deployment uses.

Logical types supported in the Information Store

The following logical types are supported in the Information Store:

- Boolean
- Date
- Date and Time
- Decimal
- Double
- Geospatial

- Integer
- Multiple Line String
- Selected From List
- Single Line String
- Suggested From List
- Time

Creating link types

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

Link types represent the possible different types of relationships that you would like to investigate between entities in your system. As such you must define the types of entity that can support each type of link.

For example, an Owner link type can create links between People, Organizations, Property, and Vehicle entities.

To create a link type:

1. In the **Schema Designer** window, click **Insert > New link Type**.

The link type has a default name, empty description field, and an ID that cannot be configured.

2. Enter a name and description for your link type.
3. Assign a suitable semantic type that sufficiently defines the link type.
4. Create property types for the link type.
5. Selecting link type end constraints is necessary to identify the possible entity types at the link's end. They are also used to create context for a search command.

a) Open the **Link Ends** tab for the selected link. All the entity types that are stored in the schema display in both the **From End Types** and **To End Types** lists.

b) Select the entity types that apply to each end of the link.

Your link end constraints are specified for your link type.

Your link type is successfully created with the specifications for its property types, semantic type, and link end constraints.

Related concepts

Schema validation

To ensure the schema functions and is safe to use in your production environment, there are restrictions your schema must comply with when you create and develop it. The number and types of restrictions that are imposed on your schema vary depending on what stage of the development it is in.

Related tasks

Deriving custom semantic types

You can derive a custom semantic type when the semantic types that are stored in the standard library do not sufficiently classify the data that they are being assigned to. You can derive a custom semantic type from an appropriate generalized parent semantic type.

Creating property types

Property types allow you to classify the data that relates to a particular item type. Create a property type for each aspect of the data that you would like to be able to analyze.

Changing link strengths

You can create link strengths to visually indicate how much confidence there is in the information that is represented by a particular link. Link strengths that you create in the schema can be applied to all link types.

Assigning semantic types

You can assign entity, link, and property types a semantic type to identify the real-world meaning of the data. Ensuring that your semantic types are assigned accurately is important because it affects the organization and interpretation of your data.

Creating labeling schemes

To define how your data is labeled on screen, or as chart item type labels, you can create labeling schemes. The labeling scheme determines what information the user sees when items are listed or visualized.

You can create item labels that determines how items are identified. For example, you create a labeling scheme for a Person entity type that has a Forename, Surname, and Date of Birth property type. You can use the Forename and Surname property types as the label.

Note: To maintain consistency, you can configure your charting schemes to use the labeling schemes to determine the chart item label.

To create a labeling scheme for your item type:

1. After you select **Labeling Scheme** in the navigation tree in the **Schema Designer** window, select the **Item type** list. The **Properties** list is updated with the property types created for your item type.
2. Select the properties to use in the label, and click the arrow to move them to the **Label parts** column.
3. Optional: Add any fixed values to the Label parts by entering them in the Free Text box and clicking the corresponding arrow. Free text can be used to separate multiple properties when combined on a label, but also provide a standard label for items that do not need a unique label

Note: For items that do not have property values to define their label, the system displays <No label> by default. Adding any character other than spaces as free text changes this behavior, displaying the free text instead.

If you want your labeling scheme to be used with your chart, you must map the labeling scheme in the charting scheme editor.

4. Optional: Rearrange the Label Parts by clicking the up and down arrows as needed.

Your labeling scheme is created and applied to your item types based on your configuration. Your labeling scheme is composed of any property types you specified to be labeled.

Semantic types and charting schemes

Analyst's Notebook Premium can interpret and use any items that are defined in a schema. However, there is additional information that can be used by Analyst's Notebook Premium to interpret data.

Semantic type information and charting schemes are used to transform data into a format that Analyst's Notebook Premium recognizes.

- Semantic types can be assigned to item, property, and property group types to identify the real life meaning in the data.
- Charting schemes determine the mapping between items in i2 Analyze and the same items when displayed on a chart in Analyst's Notebook Premium.

Assigning semantic types

You can assign entity, link, and property types a semantic type to identify the real-world meaning of the data. Ensuring that your semantic types are assigned accurately is important because it affects the organization and interpretation of your data.

Item and property types are aligned and interpreted based on what semantic type is assigned to them.

For example, a Car entity type and a Plane entity type can be assigned the Vehicle semantic type. Analyst's Notebook Premium aligns item types that are assigned the Vehicle semantic type, and interprets them as a type of vehicle.

You can assign semantic types to restrict the type of data that can be stored in a property type. Data types of property values must be consistent with the property's meaning.



Attention: If you are connected to multiple data sources, it is important to ensure that semantic types are assigned correctly because data alignment depends on it. Having common property semantic types help searches to retrieve similar items from all of the data sources.

You must assign a semantic type to a property type, unless all three of the following conditions apply:

1. It is only used as a key, label, or description.
2. It is not displayed in a report, the search results grid, or the chart data record.
3. It has no search operators defined.

To assign a semantic type to your item or property type:

1. Select the item or property type you want to assign a semantic type to.
2. In the selected item or property type's tab, click **Select** next to the semantic type field.
The **Select Semantic Type** window opens. It is the IBM i2 semantic type library that is used by Analyst's Notebook Premium.
3. Browse the library to find and select the semantic type that defines your data to the depth you require.
4. When your semantic type is identified and selected, click **Assign**.
Your item or property type is assigned a semantic type. You can remove this semantic type by clicking **Remove**.

Related concepts

Schema structure

You can determine the structure of your schema by analyzing your data in terms of item and property types. The schema structure is based on your data model.

Related tasks

Creating link types

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

Deriving custom semantic types

You can derive a custom semantic type when the semantic types that are stored in the standard library do not sufficiently classify the data that they are being assigned to. You can derive a custom semantic type from an appropriate generalized parent semantic type.

Ensuring that you choose the correct parent semantic type is important because the custom semantic type inherits characteristics and behavior from the parent. If you choose a parent semantic type that is not fitting, Analyst's Notebook Premium might interpret the data incorrectly.

For example, it would be inappropriate for you to derive a custom semantic type for a sporting event from the Document entity semantic type. A sporting event is not a specialization of a document. The result of this inappropriate specification would be that Analyst's Notebook Premium would expect to interpret documentation as opposed to sporting event data.

You can edit the details for a custom semantic type at any time, but it is best to avoid changing them in a way that changes the original meaning. Different versions of the same custom semantic type align regardless of the custom semantic type's name, description, or synonyms.



CAUTION: Defining a custom semantic type when the library already contains an appropriate semantic type can cause problems with information retrieval. Information that is retrieved from your data source would be unaligned with information retrieved from other data sources that have the correct semantic type assigned. Search carefully for an available semantic type before you create your own custom semantic types.

To derive a custom semantic type:

1. After you select the item type you want to assign a semantic type to in **Schema Designer**, click **Select** to open the **Select Semantic Type** window.
2. Locate the available semantic type that is the most similar to the specialization that you require, by searching for semantic types that have a generalized name.
3. Select the generalized type in the tree of semantic types, and then click **Derive Custom Type**. A new semantic type is added as a child of the generalized semantic type.

Note: Custom semantic types are given a special red icon to differentiate them from the standard IBM i2 semantic types.

4. Change the name of the custom semantic type to a name that reflects your usage.
5. To improve search results when you try to locate this semantic type in the future, add synonyms for this custom semantic type in the **Synonyms** text box. Separate each synonym with a comma. Phrases are allowed.
6. Describe the intended usage for this custom semantic type in the Description box. You can type a web address to provide a hyperlink to a web page for more information.
7. After your new custom semantic types are configured to your specification, click **Assign** to assign this new custom semantic type to your item type.
8. Optional: You can save the custom semantic type by clicking **File > Custom Semantic Types > Save Library**.
9. Optional: You can load a library of custom semantic types by clicking **File > Custom Semantic Types > Load Library**

Note: If you choose to save to a file name that exists, the content of the library is overwritten. This library can be merged into other schemas to use the custom semantic types it contains.

Your custom semantic type is added to your library.

Related tasks

Creating link types

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

Charting schemes

Charting schemes determine the mapping between items in i2 Analyze and the same items when displayed on a chart in Analyst's Notebook Premium. You can transform data in different ways by creating multiple charting schemes.

Property types from your schema can be mapped to chart item property types and attribute classes.

Mapping your property types to chart item property types correctly is important because they define how property values from i2 Analyze are represented on a chart.

The effectiveness of a charting scheme depends on how accurately defined your schema is. It is important to ensure that your schema is configured and detailed to meet the requirements of your organization before you create a charting scheme that will match it.

Related concepts

Schemas

A schema defines the structure of data that will be stored within your system. A schema must be designed in a way that effectively maps the data that you have, to the type of analysis that is going to be carried out.

Charting scheme validation

When you create or edit any charting scheme in Schema Designer, it is validated upon creation or when edited. Charting schemes that are not created within Schema Designer might contain invalid information that does not coincide with the configuration of your schema.

When you open a charting scheme that is not created in the Schema Designer, the following validation checks occur:

- Whether the external charting scheme's underlying XML is valid.
- Whether the XML has definitions that match the definitions that are found in the schema. If the definitions are different, a log of errors appears and prevents the external charting scheme from opening. The charting scheme can open when the prompted errors are corrected.
- Whether the logical types that are defined in the charting scheme match the logical types that are specified in the schema. If the definitions do not match and you attempt to open the charting scheme, the charting scheme opens with a log of errors. For the charting scheme to function properly, you must correct the errors that are mentioned in the log.

Related tasks

Creating charting schemes

You can create charting schemes to determine how items that are stored in i2 Analyze are mapped to Analyst's Notebook Premium items. The representation of your data depends on how your accurately your schema and charting scheme is configured.

Creating charting schemes

You can create charting schemes to determine how items that are stored in i2 Analyze are mapped to Analyst's Notebook Premium items. The representation of your data depends on how your accurately your schema and charting scheme is configured.

If you use the Schema Designer to create a new charting scheme from an existing schema, it is populated with 4 charting schemes. Each charting scheme contains mappings for all the item types from that schema. Within the entity and link type entries, there is a default entry.

You can configure the default entry for entity and link types. To apply your default entry to the item types in your schema, they all must have property types with the same name. The default configurations are used unless a type-specific mapping exists. For example, if your Default entry is configured and a specific Person type mapping is inserted, then your Account uses Default, and Person uses Person.

You can choose to create data records on the mapped chart items. The data records contain the information in the i2 Analyze item, regardless of whether it is mapped or not. You can also choose to include property gradings in the data record.

To create or edit a charting scheme:

1. In the **Schema Designer** window, click **File > Edit Charting Schemes**.

If your current schema has no associated charting schemes, a window opens with four default charting schemes in a navigation tree. If your schema has charting schemes that are associated with it, a window opens with the charting schemes in the navigation tree.

In the **Edit Charting Scheme** window you can create, delete, and edit charting schemes.

2. When you are finished creating, deleting, or editing charting schemes, you can click OK to save your changes. Alternatively, you can click Cancel if you do not want to save any creations or changes.

Related concepts

Charting scheme validation

When you create or edit any charting scheme in Schema Designer, it is validated upon creation or when edited. Charting schemes that are not created within Schema Designer might contain invalid information that does not coincide with the configuration of your schema.

Inserting chart item property types

You can map the property types in your schema to an Analyst's Notebook Premium chart by inserting chart item property types into a charting scheme. Chart item property types determine how the property values that are stored in i2 Analyze, are represented on the chart.

The **Edit Charting Schemes** window must be open. Select the item type that you want to create a property mapping for.

The following are chart item property types that are available for mapping to include descriptive pieces of information you might want visualized on the chart:

- Label
- Date
- Time
- Description of date & time
- Description
- Source reference
- Source type

In the chart item property type configuration screen, you can insert multiple property types to map to the chart item property type. Multiple chart item properties can be mapped to by the same i2 Analyze property.

You can configure whether the mapped property types have a prefix or suffix for their i2 Analyze property values. You can also determine how you want the i2 Analyze properties to behave if they have multiple values.

Labeling schemes for chart item property types are created to label the data that is represented on a chart for analysis. To map from the label of an item in i2 Analyze, you must specify which labeling scheme you want to use.

To map a property type to a chart item property type:

1. Right-click on the **Properties** entry of the relevant property type, and select **Insert Chart Item Property Type**.

The **Add Chart Item Property Type** window is shown.

2. From the **Available chart item property type** list, select a property and click **OK**. The Available chart item property type list contains any chart item property type that is not mapped.

The appropriate property type page is shown, for example Label.

3. Configure the property type's specifications to how you want them to be represented on the chart. The property type setup depends on the property type that is selected. You can define a label from as many property types as you want, and include spaces or new lines. For example, given name followed by a space followed by surname.

Your property type is successfully mapped to a chart item property.

Multiple value settings

It is possible for an item in i2 Analyze to contain properties with multiple values. Charting schemes contain configurable specifications that define how multiple property values are charted.

For example, an item might have several properties of type Middle name, all of which have different values. Items on a chart do not support this notion, so you must create a charting scheme and configure it to accommodate the multiple property values.

You can configure the property value specifications when you insert property or attribute mappings.

The configuration options for determining the value that is charted depends on the underlying data type of the property. For example, you can chart a Date & Time property value by using the earliest or the most recent value from the set of multiple values.

Note: The default configuration is set to **Use any single value**. For the default configurations, they hold no deterministic rules that specify which value in the set of values is charted.

When links are summarized - Single, Directed or Flow - each link item that is being summarized might have multiple values for a property. When you are creating link type property mappings, the following extra options are available for populating the set of values from which the charted value is selected:

- All values from all links
- Primary property from each link



CAUTION: The Information Store and Analyst's Notebook Premium do not support the concept that a single property type can possess multiple property values. Data loss can occur if you create and upload a charting scheme that accommodates for multiple property values, and place items with selected property values onto a chart.

Related tasks

Configuring link summarizations

If there are multiple links between the same two entities on an Analyst's Notebook Premium chart, you can choose how to represent those links to compact information. Your links can summarize information such as transaction links with the Flow option selected, and can show the transactions between two bank accounts with all of the transaction amounts.

Inserting attribute instances

You can map property types to instances of attribute classes. You can define a new attribute class to map to, or map your property type to an existing attribute class.

The **Edit Charting Schemes** window must be open. Select the item type that you want to map an instance of an attribute class to.

To insert an instance of an attribute class mapping:

1. Right-click your property type and select **Insert Attribute Instance**.
2. In the **Insert Attribute Class** window, you can either select an attribute class that exists, or create one specifically for the property type.
3. If you want to map to an existing attribute class, click **Select an existing attribute class** and choose an attribute from the list.
4. If you want to create an attribute class specifically for the property type:

a) Click **Select a repository property** and enter a name for your new attribute class in the **Specify the new attribute class name**.

b) Click **Next** and configure the attribute class.

5. When your attribute class is configured, click **OK**.

Your property type is mapped to an instance of an attribute class. If your attribute class was created specifically, it appears in your attribute classes section and your property type's attribute section.

Inserting attribute classes

Attribute classes determine the type of data that is stored and displayed as attributes on an Analyst's Notebook Premium chart. You can create attribute classes and map property types to them or store them within an Analyst's Notebook Premium chart template.

Attribute classes contain configurable options that include naming the attribute class, selecting its semantic type, and deciding how it is displayed on the chart. You can also configure how the attribute class behaves with other items on the chart with regards to pasting and merging.

Attribute classes must have unique names. If an attribute class of the same name exists in the current chart, the settings for the attribute class are retained over the settings in the charting scheme.

To create an attribute class for your Analyst's Notebook Premium chart template:

1. In the **Edit Charting Schemes** window, expand the tree structure for the charting scheme and right-click the **Attribute Classes** entry.
2. Select the appropriate menu item for the type of attribute class that you want to create: **Text**, **Flag**, **Number**, or **Date & Time**.

A new attribute class is created in the tree view.

3. Configure the details to your specifications in your New Attribute Class' window. You can configure your attribute class' name, appearance on the chart, semantic type, and behavior to use for attribute values of this class when you merge or paste chart items.
4. When your attribute class is configured, click **OK**.

Your attribute class is successfully configured and inserted. If an attribute class that is specific to a property type is inserted, the configurations for that attribute class appear when that property type is used.

Configuring link summarizations

If there are multiple links between the same two entities on an Analyst's Notebook Premium chart, you can choose how to represent those links to compact information. Your links can summarize information such as transaction links with the Flow option selected, and can show the transactions between two bank accounts with all of the transaction amounts.

To configure link summarization for a link type:

1. In the tree view of the **Edit Charting Schemes** window, expand the charting scheme that you wish to edit and click and expand **Link Types**.
2. Select the link type that you want to configure link summarization for.
3. In the **Link Summarization** area, select one of the following options.

Option	Description
Single Link	All the links of the selected link type between the same two entities are combined into a single link. This option is useful if you are producing a summary chart and do not want to show all of the details.
Directed	All the links of the selected link type in the same direction between the same two entities are combined. This option is useful when charting information such as telephone calls or transactions.

Option	Description
Multiple	Each link of the selected link type between the same two entities is charted separately. This option is useful if you want to show all of the detail but if there are many links, it might make the chart cluttered or unreadable.
Flow	All the links of the selected link type are combined into a single link. The direction (or flow) is determined from a property that you specify. For example, if there are several financial transactions between two bank accounts, this option is useful to determine the direction that the aggregate amount of the money is flowing. Select a property from the Property used to calculate flow list. Flow value is calculated and is available for you to map in an attribute instance or chart item property mapping. Note: The Flow option is available only when the selected link type has numeric data properties.

4. When you combine multiple links between the same two entities, you can choose whether to retain the strengths by keeping links of different strengths separate. Alternatively, you can choose to use the strength of the weakest link. Select the appropriate options on the right side of the page. Your links are summarized as specified.

Related concepts

Multiple value settings

It is possible for an item in i2 Analyze to contain properties with multiple values. Charting schemes contain configurable specifications that define how multiple property values are charted.

Copying mappings

At any level in the charting scheme tree structure, you can copy the specifications of a mapping or collection of mappings to another charting scheme. You can duplicate a whole charting scheme but there might be situations when you want to copy only certain definitions in a charting scheme.

If you want to copy an element from one charting scheme to another:

1. Select the element in the tree view that you want to copy.
2. Click **Copy to**.

The **Copy To** window is shown.

3. Select the check boxes for the charting schemes that you want to copy the element to and click **OK**.

Note: When you copy an element, the **Copy to** window indicates whether a definition exists in the target scheme for the entry you want to copy. Any existing definition in the selected charting scheme is overwritten.

If you want to duplicate a charting scheme:

4. Select the charting scheme that you want to duplicate, and click **Duplicate**.

Your charting scheme is successfully duplicated and appears in the navigation tree with the prefix 'Copy'.

The definitions that are contained in the selected entry in the tree structure and all its child entries are copied to the selected charting scheme. For example, if you select Entity Types in the tree structure and click **Copy to**, all of the property and attribute mappings are also copied.

Changing the grading scheme

You can grade your properties in i2 Analyze to indicate the level of confidence you have in property values. You can create and configure grade types within a grading scheme.

Different organizations have different grading systems, and so you can create and configure new grade types, with custom grading values. Default grading types can be configured to an organization's requirements. For example, the UK law enforcement uses the following grading model:

Source evaluation

- A - Always reliable
- B - Mostly reliable
- C - Sometimes reliable
- D - Unreliable
- E - Untested source

Intelligence evaluation

- 1 - Known to be true without reservation
- 2 - Known personally to source but not to officer
- 3 - Not personally known to source but corroborated
- 4 - Cannot be judged
- 5 - Suspected to be false or malicious

Handling code

- 1 - Can be disseminated to other law enforcement and prosecuting agencies, including law enforcement agencies within the EEA, and EU compatible (no special conditions)
- 2 - Can be disseminated to UK non-prosecuting parties (authorization and records needed)
- 3 - Can be disseminated to non-EEA law enforcement agencies (special conditions apply)
- 4 - Can be disseminated within the originating agency only
- 5 - No further dissemination: refer to the originator. Special handling requirements that are imposed by the officer who authorized collection

To change the grading scheme:

1. After you select **Grade Types** in the navigation tree in the **Schema Designer application** window, click **Insert > New Grade Type**.

The new grade type is added to the **Grade Types** list on the left.

2. Enter a **Name** and a **Description** to identify the grade type.
3. For each grade value, enter a **Grade Value** and click **Add**.
4. Optional: Use the arrow buttons to change the order in which the grade values are displayed.
5. Optional: With a grade value selected, click **Remove** to delete the value.
6. Optional: Select the **Mandatory** check box to force users to select a value for this grade type when you enter data.

Your grading scheme is successfully modified.

Changing link strengths

You can create link strengths to visually indicate how much confidence there is in the information that is represented by a particular link. Link strengths that you create in the schema can be applied to all link types.

To change the link strengths that are available:

1. In the **Schema Designer** window, click **Insert > New Link Strength** or open an existing link strength in the navigation tree.
2. Enter a **Name** for the link strength.
3. Select the **Line style**.
4. Optional: Select **Set as default**.

A new link strength is created and configured to your investigation's specification.

Related concepts

Schema validation

To ensure the schema functions and is safe to use in your production environment, there are restrictions your schema must comply with when you create and develop it. The number and types of restrictions that are imposed on your schema vary depending on what stage of the development it is in.

Related tasks

Creating link types

You can create link types to define the structure for describing the relationships between entities. Link types are similar to entity types in that they contain property types.

Glossary

This glossary provides terms and definitions for the i2 Analyze software and products.

The following cross-references are used in this glossary:

- *See* refers you from a nonpreferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

A

abstract semantic type

A semantic type that only serves as the parent of other semantic types. Abstract semantic types categorize their child semantic types, but are never associated with real data.

access level

A measure of the rights that a user has to view or edit an item. Access levels are calculated separately for every user and every item. See also grant level, security dimension.

administrator

A person responsible for administrative tasks such as access authorization and content management. Administrators can also grant levels of authority to users.

alert

A message or other indication that signals an event or an impending event that meets a set of specified criteria.

alert definition

The statement of criteria that trigger an alert.

aligned value

A value that is used to interpret equivalent native values from different data sources. For example, the value Male can be used to align the native values M or Ma.

analysis attribute

A characteristic or trait pertaining to a chart item. Analysis attributes are never displayed on charts.

association chart

A chart that highlights the relationships between entities, rather than a chronology of events, by arranging data in a manner that emphasizes particular associations.

attribute

A piece of information that is associated with a chart item, such as a date of birth or an account number. An attribute is represented by a symbol, or a value, or both, that is displayed with the chart item.

attribute class

A descriptor of the characteristics of an attribute, including the type of its values, how its values are displayed, and the treatment of its values when they are merged or pasted on a chart.

attribute entry

An attribute with a preset value that can be associated with a chart item.

attribute instance

A single use of an attribute on a chart item.

audit

To record information about database or instance activity by applications or individuals.

authority

A measure of how well-connected an entity is, based on its inbound links. Authority is one of two eigenvector centrality measures used in social network analysis. See also [centrality](#), [eigenvector](#).

automatic attribute

An attribute that is created automatically by the application and added to a chart item.

B**betweenness**

A measure of how important an entity is, based on the number of paths that pass through it on an association chart. Betweenness is one of the centrality measures used in social network analysis. See also [centrality](#), [gatekeeper](#).

binding strength

A measure of the strength of a relationship between two entities that are directly or indirectly linked. See also [common neighbor](#).

box

An entity representation that can indicate an organization or group on a chart. A box is often used to enclose other entities. See also [circle](#), [representation](#).

C**card**

A record of information attached to an item. An item can have multiple cards.

centrality

The relative importance of one entity compared to other entities in social network analysis, as determined by its relationships. See also [authority](#), [betweenness](#), [closeness](#), [degree](#), [eigenvector](#), [hub](#), [social network analysis](#).

chart

A visual representation of real-world objects, such as organizations, people, events, or locations, and the relationships between them.

chart fragment

A view of a chart that highlights particular items of interest.

charting scheme

A definition that describes how item data behaves when it is visualized on a chart. For example, how data is copied into chart item properties, the chart template and labeling scheme to use, and whether to display attributes and pictures. See also [chart template](#).

chart property

A characteristic of a chart, such as its summary description, time zone, grid size, background color, or merge and paste rules. Chart properties are saved with the chart. See also [chart template](#).

chart template

An object that is used for chart creation that contains preconfigured chart properties, and lists of permitted entity types and link types. See also [chart property](#), [charting scheme](#).

child

In a generalization relationship, the specialization of another element, the parent. See also [parent](#).

circle

An entity representation that can indicate an organization or a group on a chart. A circle is often used to enclose other entities. See also [box](#), [representation](#).

circular layout

A layout in which entities are arranged by type around the circumference of a circle. See also [layout](#).

cloaked item

An item whose existence is known to the user but whose information is hidden from the user. See also [item](#), [placeholder](#), [signpost message](#).

closeness

A measure of how quickly an entity can use links to get access to other entities on an association chart. Closeness is one of the centrality measures used in social network analysis. See also [centrality](#).

cluster

A group of entities that have more connections to each other than to entities outside the group.

common neighbor

An entity that is directly connected to at least two other entities. For example, if C is connected to A and B, then C is a common neighbor of A and B. See also [binding strength](#), [connection](#).

compact peacock layout

A layout in which complex groups of linked entities are arranged to highlight the structure of associations. It is most suitable for charts with many linked entities. See also [layout](#).

condition

A specified property, a value, and an operator that defines a comparison relationship between them. One or more conditions can be used to create a query or a conditional formatting specification. See also [parameterized query](#).

conditional formatting

The process of defining and applying rules to change the appearance of chart items automatically, based on their properties. See also [conditional formatting specification](#).

conditional formatting specification

A collection of conditional formatting rules. See also [conditional formatting](#).

connection

A direct relationship between a pair of entities on a chart, represented by one or more links. See also [common neighbor](#), [connection multiplicity](#), [directed connection](#).

connection multiplicity

A setting that controls whether multiple links between the same items are displayed as a single line, as directed lines, or as multiple lines. See also [connection](#).

controlling item

A chart item whose position on the chart is defined by its date and time, and whose position affects the positions of other timed items. See also [free item](#), [ordered item](#).

cover sheet

A page on which the user can view and edit the summary and custom properties of a chart.

D**degree**

A measure of how many direct relationships an entity has with other entities on an association chart. Degree is one of the centrality measures used in social network analysis. See also [centrality](#), [root entity](#).

directed connection

A connection between entities in which links that are in the same direction are represented as a single link on a chart. See also [connection](#).

diverted theme line

A theme line that is attached to an event frame such that when the event frame is moved, the theme line maintains its vertical position with respect to the frame. See also [event frame](#), [theme line](#).

E**eigenvector**

A measure of how well-connected an entity is, based on its inbound and outbound links. Eigenvector is one of the centrality measures used in social network analysis. See also [authority](#), [centrality](#), [hub](#).

end

An entity that is attached to a link. See also [end constraint](#).

end constraint

A constraint on the types of entities that can be the end of a particular link. See also [end](#), [valid end type](#).

entity

A set of details that are held about a real-world object such as a person, location, or bank account. An entity is a kind of item.

entity semantic type

A semantic type that can be assigned only to an entity or an entity type. See also [semantic type](#).

entity type

A descriptor of the characteristics of an entity, including the properties it can contain and its appearance in visualizations.

event frame

An entity representation that emphasizes date and time information. An event frame is often used in conjunction with theme lines. See also [diverted theme line](#), [representation](#).

excluded word list

A list of words that are ignored when they are entered as search terms.

expansion

A process that searches for entities within a data source that are directly related to some selected entities.

F**free item**

A chart item that is not ordered. Free items can be moved anywhere on the chart. See also [controlling item](#), [ordered item](#).

G**gatekeeper**

An entity with a high measure of betweenness that may control the flow of information among other entities on an association chart. See also [betweenness](#).

grade

A rating that indicates the accuracy of a piece of information or the reliability of an intelligence source.

grading system

A rating scale that is used to classify information in a data store or on a chart. A grading system is a measure of reliability and accuracy.

grant level

A measure of the rights that a user has to change the security permissions of an item. Grant levels are calculated separately for every user and every item. See also [access level](#), [security dimension](#).

grouped layout

A layout in which entities are arranged to show groups of interconnected entities. See also [layout](#).

H**heat map**

A graphical representation of data values in a two-dimensional table format, in which higher values are represented by darker colors and lower values by lighter ones.

hierarchical layout

A layout in which entities are arranged to show organizational structures. See also [layout](#).

histogram

A graphical display of the distribution of values for a numeric field, in the form of a vertical bar chart in which taller bars indicate higher values. See also [histogram filter](#).

histogram filter

A filter that changes the appearance of a chart. When a histogram bar is selected, items that match the conditions defined by that bar are selected, while items that do not are hidden. See also [histogram](#).

hub

A measure of how well-connected an entity is, based on its outbound links. Hub is one of two eigenvector centrality measures used in social network analysis. See also [centrality](#), [eigenvector](#).

I**icon**

An entity representation that consists of a stylized image and an optional label. See also [representation](#).

import design

A specification of how data from an external source will be transformed into chart items during an import procedure.

item

An entity or a link. Items are characterized by the values of their properties. See also [cloaked item](#), [merged item](#), [ordered item](#).

L**labeling scheme**

A specification for combining property values to be displayed on screen, or as chart item labels.

layout

The arrangement of items on a chart. See also [circular layout](#), [compact peacock layout](#), [grouped layout](#), [hierarchical layout](#), [minimize crossed links layout](#), [peacock layout](#).

line strength

An indication of confidence in the information underlying a particular link. Line strength is represented as a solid, dashed, or dotted line on a chart.

link

An association between two entities, such as an ownership relationship between a person and a vehicle.

link direction

An indication that the meaning of a link is different for each of its ends. For example, the direction of a telephone call makes one end the caller and the other the recipient. Link direction can influence the centrality measures used in social network analysis.

link semantic type

A semantic type that can be assigned only to a link or a link type. See also [semantic type](#).

link separation

The distance between adjacent links in a connection on a chart.

link type

A descriptor of the characteristics of a link, including the properties it can contain and its appearance in visualizations.

M**match**

The part of a result that met a condition during a search operation. A search can yield a perfect match or a partial match.

merged item

An item that is created by merging the information held in two or more items. See also [item](#).

minimize crossed links layout

A layout in which entities are arranged in a configuration where the fewest number of links overlap. See also [layout](#).

multiplicity

See [connection multiplicity](#).

N**network chart**

See [association chart](#).

O**ordered item**

A chart item whose position is maintained within a sequence. The movement of an ordered item is restricted such that it cannot be dragged beyond neighboring ordered items. See also [controlling item](#), [free item](#), [item](#).

P**parameterized query**

A query with conditions in which one or more parameters are defined. The parameter values are set by the user. See also [condition](#).

parent

In a hierarchy or auto-level hierarchy, a member that has one or more child members at the level immediately below.

path

A route on a chart between two entities. A path may include intermediate entities.

peacock layout

A layout where complex groups of linked entities are arranged to show the structure of associations. It is most suitable for charts with many linked entities. See also [layout](#).

pick list

A data category that has a limited number of permissible values, which are often presented in a drop-down list in the user interface.

placeholder

A redacted version of an item that is displayed to the user in situations where displaying the full item is not possible or not permitted. See also [cloaked item](#), [signpost message](#).

property

A container for a single piece of information about an item.

property group

A piece of information about an item that comprises related properties. For example, a distinguishing feature of a person comprises information about the type, appearance, and location of the distinguishing feature.

property semantic type

A semantic type that can be assigned to a property type, a property in a data record, or an attribute class. See also [semantic type](#).

property type

A descriptor of the characteristics of a property, including the type of information it can contain.

proportional

Pertaining to an area of a chart in which the horizontal distances between items have a linear relationship with the time differences between them.

R**representation**

The form in which an entity is represented on a chart. See also [box](#), [circle](#), [event frame](#), [icon](#), [theme line](#).

root entity

An entity in a grouped layout that has the highest degree centrality in its group. Depending on the data, there can be more than one root entity. See also [degree](#).

S**schema**

A complete description of all the entity types, link types, and their associated property types that are available for items within a system.

security dimension

A collection of related values that can be used to label a user according to their role or security clearance, with the aim of affecting their access to information. See also [access level](#), [grant level](#).

semantic type

A category that defines the real-world meaning of data, and therefore how applications should interpret that data. For example, Person is a semantic type that could be assigned to entity types such as Male, Victim, and Witness. See also [entity semantic type](#), [link semantic type](#), [property semantic type](#).

signpost message

A piece of text that is stored and displayed with an item or a placeholder. The signpost message explains how to obtain more control over, or more information about, the item. See also [cloaked item](#), [placeholder](#).

snapshot

A stored version of a chart that preserves its contents and layout at a particular stage of its development.

social network analysis

A method of analyzing the structure of social relationships that uses mathematical metrics to make claims about social organization and social dynamics. See also [centrality](#), [weight](#).

source reference

An identifier that indicates the source of information, for example, a document reference number.

style segment

A section of a theme line between adjacent items to which color and strength can be applied.

T**theme line**

An entity representation that shows the interactions of an entity over time. A theme line can be used with event frames. See also [diverted theme line](#), [representation](#).

theme line extent

The distance between the beginning and end of a theme line

theme line wiring

The manner in which a theme line diverts from a horizontal trajectory in order to pass through and travel between event frames.

timeline chart

A chart or a portion of a chart that shows a chronology of events. For example, a series of meetings that occur over several days, or a set of transactions that occur over a period of time.

V**valid end type**

An entity type that conforms to the end constraints of a particular link. See also [end constraint](#).

W**weight**

A value that is added to a link on an association chart, to represent its importance relative to other links. Weight can influence the centrality measures used in social network analysis. See also [social network analysis](#).

weightings file

A file that contains information that can apply weighting values to links on a chart.

wiring segment

The section of a theme line between adjacent diverting event frames.

Managing data in i2 Analyze

In a running deployment of i2 Analyze, and subject to their permissions, users are able to add and remove data from the Information Store. As an administrator of the system, you have access to tools for ingesting and deleting data from the store in bulk, and also for alerting users when something in the system changes.

Troubleshooting and support

[i2 Enterprise Insight Analysis support page](#)

[IBM Software Support home page](#)

Ingesting data into the Information Store

System administrators can add data to the Information Store in bulk by following an ETL (extract, transform, load) process. To update data that was added to the Information Store earlier, they can use the same process to add it again.

Ingesting data

[Information Store data ingestion](#)

To enable its analytical capabilities, i2 Analyze places requirements on the data records that it processes. i2 Analyze records must conform to the general data model and a deployment-specific i2 Analyze schema, and they need information that might not be present in external sources. Before you can add data from your source to the Information Store, you must transform it to meet the same requirements.

[Information Store data correlation](#)

This documentation provides an overview of the correlation functions that are available during data ingestion into the Information Store in IBM i2 Analyze. Later sections describe how to configure correlation with an example use case.

[Information Store data deletion](#)

More information

[IBM Security Skills Gateway](#)

[IBM training](#)

Information Store data ingestion

To enable its analytical capabilities, i2 Analyze places requirements on the data records that it processes. i2 Analyze records must conform to the general data model and a deployment-specific i2 Analyze schema, and they need information that might not be present in external sources. Before you can add data from your source to the Information Store, you must transform it to meet the same requirements.

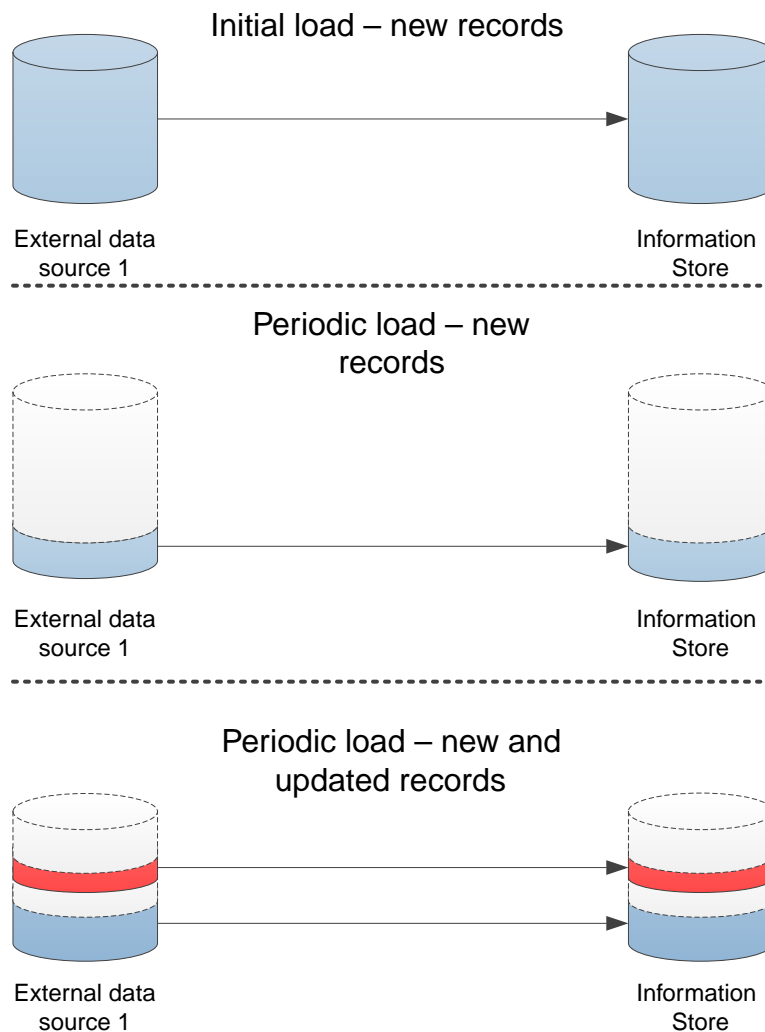
i2 Analyze provides two mechanisms that make it easier to align your data with an i2 Analyze schema and prepare it for ingestion:

- The i2 Analyze deployment toolkit includes a command for creating staging tables that match the structure of the schema, in the same database as the Information Store. If you can load your data into the staging tables successfully, the Information Store can ingest your data from those tables.
- During data ingestion, i2 Analyze uses ingestion mappings that describe how to create records in the Information Store from the data in the staging tables. You can define these mappings in one or more files.

There are two scenarios for ingesting data from an external source. The first time that you load data of a particular item type from an external source, you perform an *initial load*. In an initial load, you are presenting the data to the Information Store for the first time. After you perform the initial load,

you might want to perform *periodic loads* that update the Information Store as data is added to the external source or the data is changed. The same staging tables and mapping files can be used in both scenarios.

The following diagram shows the types of data that can be included in each type of load that you might perform. The data in blue is presented to the Information Store for the first time, and the data in red contains updates to data that exists in the Information Store.



The i2 Analyze deployment toolkit provides two *import modes* that you can use to ingest data:

Bulk import mode

The *bulk* import mode can be used in the following situations:

- Initial load, new records, no correlation
- Periodic load, new records, no correlation

The bulk import mode is significantly faster for ingesting new, uncorrelated records.

Standard import mode

The *standard* import mode can be used in the following situations:

- Initial load, with or without correlation
- Periodic load, new and updated records, with or without correlation

Standard import mode trades some performance for increased data validation and additional operations during ingestion.

For more information about correlation, see [Overview of correlation](#).

Because ingestion is completed on a per-item-type basis, you can use the bulk import mode for some item types and use standard mode for others from the external source. You can also use the bulk import mode to complete an initial load, and then use the standard import mode to complete periodic loads to update the data.

About this guide

This documentation explains how to add data from external sources to the IBM i2 Analyze Information Store, and how to keep the Information Store up-to-date as the contents of the external source changes. The requirements are mostly the same in both scenarios, but there are some differences depending on the data that you are updating the Information Store with.

Intended audience

This guide is intended for users who understand both the [i2 Analyze data model](#) and the [i2 Analyze schema](#) that defines the data structure of their deployment. Populating the Information Store staging tables also requires you to be familiar with your database management system.

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users to resolve problems with the product.

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM software subscription and support contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

To contact IBM Support about a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM Support topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
 - Online through the IBM Support Portal at <https://www.ibm.com/mysupport/>. You can open, update, and view all of your service requests from the Service Request portlet on the Service Request page.
 - By phone. For the phone number to call in your region, see the Directory of worldwide contacts web page at <https://www.ibm.com/planetwide/>

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support

website daily, so that other users who experience the same problem can benefit from the same resolution.

Information Store ingestion and the production deployment process

Before you deploy i2 Analyze into production you develop your configuration in a number of environments. The same is true for the ingestion process. You use the same environments to develop the Information Store ingestion process for each of your external data sources.

In the *configuration development environment*, ingest small amounts of test data that is representative of the entity and link types in your existing data sources. By doing this, you can learn how to ingest your data into the Information Store database and ensure that your i2 Analyze schema and security schema are correct for your data. If you discover that the schemas cannot represent your data, you can update your schema files.

You are likely to complete many test ingestions during this development phase. As you understand the process more, you might change how the data is loaded into the staging tables, the data that you use to generate identifiers, or change the ingestion mappings. If you modify the i2 Analyze schema, you must re-create your staging tables to match.

The ingestion mappings that you use are slightly different for each external data source that you ingest data from. It is recommended that you keep any ingestion mapping files in a source control system to track changes and so that you can use them in later development and production environments.

In your *pre-production environment* or *test environment*, use the process that you developed to ingest larger quantities of data and note the time that is taken to ingest the data. You can complete representative initial and periodic loads from each of your external data sources. The time that each ingestion takes allows you to plan when to ingest data in your production environment more accurately.

In your *production environment*, complete your initial loads by using the developed and tested process. According to your schedule, perform your periodic loads on an ongoing basis.

Your deployment architecture can impact the ingestion process in general, and any logic or tools used for transformation of data. For more information, see [“Understanding the architecture”](#) on page 387.

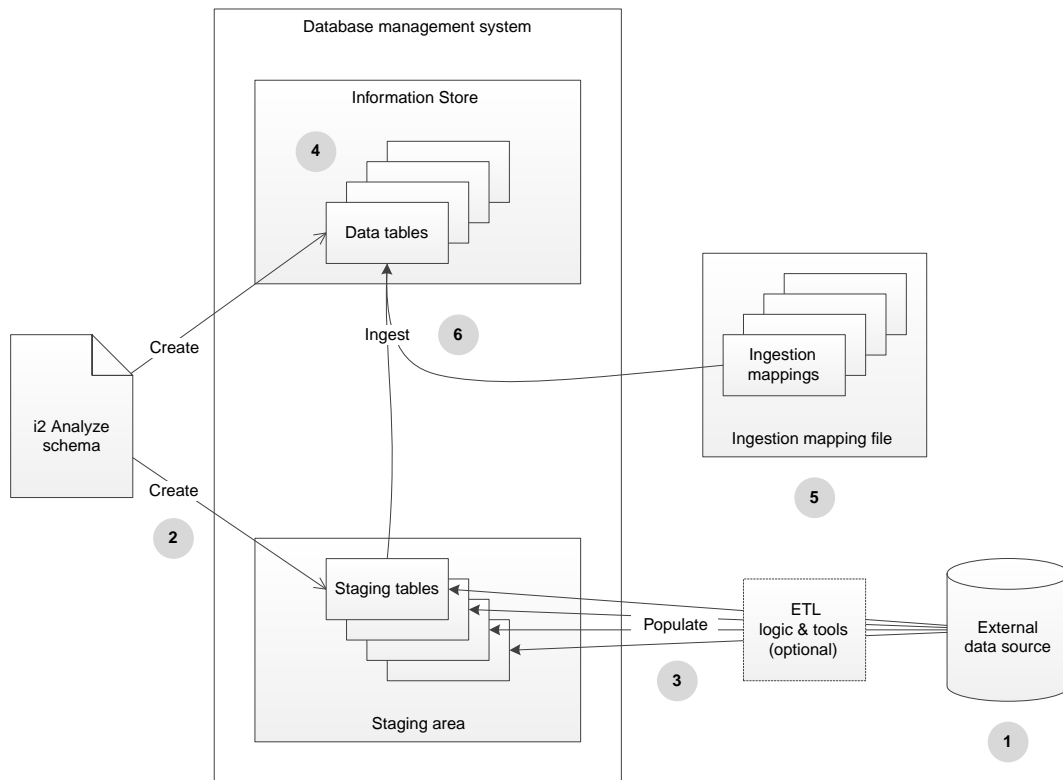
The instructions that describe the ingestion process are separated into [preparing for ingestion](#) and [adding data to the Information Store](#).

Preparing for ingestion

You must complete three tasks before the Information Store can ingest data from an external source. You must identify exactly which data to load, transform the data to align with the active i2 Analyze schema, and augment the data with extra information that the Information Store requires.

The only way to add and modify large volumes of data in the i2 Analyze Information Store is to enable and then instruct the Information Store to ingest it. The enablement process involves creating and populating staging tables for the data, and then supplying the metadata that is crucial to the analytical capabilities of i2 Analyze.

You can plan and run the Information Store data ingestion process in a series of discrete steps. This diagram illustrates the approach.



1. Decide which entity types and link types in the active i2 Analyze schema best represent the data that you want the Information Store to ingest.
2. Create staging tables in the database for the types that you identified. Create more than one staging table for some link types.
3. Use external tools, or any other appropriate technique, to transform your data and load the staging tables with the data for ingestion.
4. Add information about your data source to the list of ingestion sources that the Information Store maintains.
5. Write the ingestion mappings that govern the ingestion process and provide additional information that the Information Store requires.
6. Run the ingestion command separately for each of the ingestion mappings that you wrote.

The `examples\data\law-enforcement-data-set-1` and `\signal-intelligence-data-set-1` directories in the deployment toolkit contain files that i2 Analyze uses when you run the `setup -t ingestExampleData` command to populate the Information Store during deployment. These files provide demonstrations of many of the steps in the standard approach to ingestion. The following topics describe those steps in more depth as they detail the Information Store ingestion process.

Identifying the data to be ingested

The detail of how you arrange for the Information Store to ingest your data varies according to how that data is stored in its source. However, the start of the process is always to consider what data you have, and work out how you can shape it to fit the i2 Analyze schema.

Usually, when you start thinking about adding data from an external source into the Information Store, an i2 Analyze deployment is already in place. That deployment necessarily has an i2 Analyze schema that defines all of the entity types, link types, and property types that data in the system can have. Before you go any further, you must have a clear idea of how your data becomes i2 Analyze entity records and link records in the Information Store.

It is unlikely that the data in your external source has a one-to-one mapping with the entity types and link types in the i2 Analyze schema:

- Probably, your source does not contain data for all the entity types in the schema. As a result, you do not usually need to create a staging table for every possible entity type.
- The schema can define link types that connect several different entity types. In that case, each entity-link-entity type combination for which your source contains data requires a separate staging table.

For example, imagine an i2 Analyze schema that defines the entity types "Person", "Vehicle", and "Account", and the link type "Access to". In this situation, you might decide to create a staging table for each of the entity types. However, the data requires two staging tables for "Access to" links: one for links between people and vehicles, and the other for links between people and accounts.

1. Open the schema for the i2 Analyze deployment in Schema Designer.
2. Go through the list of entity types, and determine which of them represent data in your source.
3. Make a note of the identifier of each entity type that represents your data.
4. Repeat steps 2 and 3 for the list of link types. Check the Link Ends tab, and make a note of all the combinations for which your source contains data.

When you complete the steps, you have a list of all the i2 Analyze schema types that your data contains. You also have a list of all the staging tables that you need to create.

Creating the staging tables

The Information Store does not ingest data directly from your data source. Instead, ingestion takes place from staging tables that you create and populate. This abstraction makes it easier for you to align your data with the Information Store, and allows i2 Analyze to validate your data before ingestion.

The staging tables that you use to ingest data into the Information Store must conform to a specific structure. For more information about the staging table structure, see [“Information Store staging tables”](#) on page 392.

The simplest approach to Information Store ingestion is to create a staging table for every entity type, and every entity-link-entity type combination, that you identified in your data. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for creating one staging table at a time.

The deployment toolkit command looks like this:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=type_identifier
      -p databaseSchemaName=staging_schema
      -p tableName=staging_table_name
```

While the ETL toolkit command looks like this:

```
createInformationStoreStagingTable
  -stid type_identifier
  -sn staging_schema
  -tn staging_table_name
```

In both cases, *type_identifier* is the identifier of one of the entity types or link types from the i2 Analyze schema that is represented in your data source. *staging_schema* is the name of the database schema to contain the staging tables. (If you are using Db2, the command creates the database schema if it does not exist. If you are using SQL Server, the schema must exist.) *staging_table_name* is the name of the staging table itself, which must be unique, and must not exceed 21 characters in length.

Important: Many of the commands that are associated with the ingestion process modify the database that hosts the Information Store. By default, the commands use the database credentials that you specified during deployment in the `credentials.properties` file.

To use different credentials in the deployment toolkit, add `importName` and `importPassword` parameters to the list that you pass to the command. To use different credentials in the ETL toolkit, modify the `DBUsername` and `DBPassword` settings in the `Connection.properties` file.

1. If you are using the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory. If you are using the ETL toolkit, navigate to the `etltoolkit` directory.
2. For each entity type or link type that you identified for ingestion, run the **createInformationStoreStagingTable** command.

For example:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=ET5 -p databaseSchemaName=IS_Staging
      -p tableName=E_Person
```

By convention, you create all of the staging tables for the same source in the same database schema, which has the name `IS_Staging` in this example. It is also conventional to name the staging table itself similarly to the display name of the entity type or link type to which the table corresponds. In this case, the staging table is for the Person entity type.

Note: When the i2 Analyze schema allows the same link type between several different entity types, create separate staging tables for each combination:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=LAC1 -p databaseSchemaName=IS_Staging
      -p tableName=L_Access_To_Per_Acc
```

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=LAC1 -p databaseSchemaName=IS_Staging
      -p tableName=L_Access_To_Per_Veh
```

This example illustrates an Access To link type (with identifier LAC1) that can make connections from Person entities to Account entities, or from Person entities to Vehicle entities. The commands create staging tables with different names based on the same link type.

At the end of this procedure, you have a set of staging tables that are ready to receive your data before ingestion takes place. The next task is to make your data ready to populate the staging tables.

Preparing the external data

The staging tables that you create during the ingestion process have data structures that are similar to, but simpler than, the Information Store data tables. Whatever your source is, you must find a way to shape the data that it contains into a form that is compatible with the staging tables.

After you create the staging tables, you can view them in IBM Data Studio (or similar software) to see the definitions of their columns. You must make your data matches these definitions before you can go on to populate the staging tables.

Because all data sources and many i2 Analyze schemas are different, there is no single procedure that you can follow to prepare your data for ingestion. However, there are a number of common considerations.

- Each staging table can contain data that maps to only one entity type or link type. If your source data has rows or records that contain data for more than one of the types that you identified, then you must separate them during preparation or population.

For data in a relational source, this preparation might mean creating views on the original tables. If the data is in CSV files, then you might need to wait until you populate the staging tables to change its shape in this way.

- The Information Store does not support storing properties with multiple values in the same i2 Analyze record. The records that you create must contain values for a maximum of one property with each permitted property type.
- If you are dealing with date and time data, that data must meet extra requirements before the Information Store can ingest it. To retain information unambiguously, the staging tables use four columns to represent date and time data.

Even if you know that your date and time data was recorded in Coordinated Universal Time, you must make that fact explicit in the data to be ingested. For example, if your source contains information about an event that started at 9 AM on October 6 2002, then the values you need to prepare are:

```
2002-10-06 09:00:00 (the data and time originally entered)
UTC (the timezone)
0 (Daylight Saving Time is not in effect)
2002-10-06 09:00:00 (the date and time in Coordinated Universal Time)
```

- The `source_id`, `origin_id_type`, `origin_id_keys` columns of the staging table are used to store values that reference the data in its original source and can be used to make up the origin identifier of the resulting record.

Note: If the staging table definition was for a link type, it would also contain `from_` and `to_` variations of each of the columns.

For more information about generating origin identifiers during ingestion, see [“Origin identifiers” on page 403](#).

- If your external source is a relational database, you might find that the only data for some links is the presence of a foreign key relationship between two tables. In that case, you must synthesize a reproducible reference for the link from the other data that you have available.

For example, you might be able to create a unique reference for a link by combining the identifiers of the entity records at its ends.

- All staging tables contain a `source_last_updated` column that you can use to store information about when the data to be ingested was modified in its source.
- All staging tables contain columns for each of the access dimensions that the security schema defines. If your external source includes security information, then you can map that information

to the security schema of your target deployment, and populate the staging table columns accordingly.

Alternatively, you can leave the security columns blank, and provide security dimension values on a mapping- or source-wide basis later in the ingestion process.

- All staging tables contain `correlation_id_type` and `correlation_id_key` columns. To correlate data that is ingested into the Information Store, use these columns to store the values that comprise the correlation identifier for each row of data. If you do not want to use correlation, leave the columns blank.

If you specify values for a correlation identifier, then also specify a value for the `source_last_updated` column, which is used during the correlation process.

For more information about correlation, correlation identifiers, and the impact of the `source_last_updated` value, see [Overview of correlation](#).

- The columns named `source_ref_source_type`, `source_ref_source_location`, and `source_ref_source_image_url` are used to populate the source reference that is generated when the data is ingested.

For more information about implementing source references in your deployment, see [Configuring source references](#).

- The staging tables for link types contain a column for the direction of the link.

The Information Store considers links to go "from" one entity "to" another. The direction of a link can be WITH or AGAINST that flow, or it can run in BOTH directions, or NONE.

- If your link data includes direction information, then you can add it to the staging table during the population process, and then refer to it from the mapping file.
- If your link data does not include direction information, then you can specify a value in the mapping file directly.

By default, if you have no direction information and you do nothing in the mapping file, the Information Store sets the direction of an ingested link to NONE.

Important: The Information Store places limits on the ranges of values that properties with different logical types can contain. If you attempt to use values outside these ranges, failures can occur during or after ingestion. For more information, see ["Information Store property value ranges" on page 395](#).

The `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit contains a set of CSV files that were exported from a relational database.

In files like `event.csv`, you can see date and time data that meets the requirements of the staging tables. You can also see multiple files for "Access to" links, and how some staged link rows contain little more than a set of identifiers.

Populating the staging tables

The i2 Analyze deployment toolkit and the ETL toolkit create the staging tables for data ingestion in the same database as the Information Store data tables. After you prepare your data, but before you can instruct the Information Store to ingest it, you must populate the staging tables.

The approach that you take to populate the staging tables depends on the database management system that you are using, the form that your source data is in, and the tools that you have available.

Db2 provides the `ingest`, `import`, and `load` utilities:

- If your data is in comma-separated value (CSV) files, then you can use the **IMPORT** or **INGEST** commands.

- If your data is in the tables or views of another database, then you can use the **IMPORT**, **INGEST**, or **LOAD** commands.

SQL Server provides the bulk insert and insert utilities:

- If your data is in comma-separated value (CSV) files, then you can use the **BULK INSERT** command.

Note: To use the **BULK INSERT** command, the user that you run the command as must be a member of the *bulkadmin* server role.
- If your data is in the tables or views of another database, then you can use the **INSERT** command.
- You can use SQL Server Integration Services as a tool to extract and transform data from various sources, and then load it into the staging tables.

Alternatively, regardless of your database management system, you can use IBM InfoSphere DataStage as a tool for transforming your data and loading it into the staging tables. You can specify the database schema that contains the staging tables as the target location for the ETL output.

The subdirectories of the `examples\data` directory in the deployment toolkit all contain a `db2` and a `sqlserver` directory.

If you are using Db2, inspect the `LoadCSVDataCommands.db2` file in the `db2` directory. In each case, this file is a Db2 script that populates the example staging tables from the prepared CSV files. The script calls the **IMPORT** command repeatedly to do its work. In most instances, the command just takes data from columns in a CSV file and adds it to a staging table in a Db2 database schema.

If you are using SQL Server, inspect the `LoadCSVDataCommands.sql` file in the `sqlserver` directory. In each case, this file is an SQL script that populates the example tables from the prepared CSV files. The script calls the **BULK INSERT** command repeatedly to do its work. The **BULK INSERT** command uses `.fmt` format files, which are also in the `sqlserver` directory, to instruct SQL Server how to process the CSV files into the staging tables. For more information about format files, see [Non-XML Format Files](#).

Defining an ingestion source

The Information Store keeps a list of all the sources from which it has ingested data. Before it can ingest data, you must tell the Information Store about your source. In the ingestion mapping file, you then specify the data source name in the mapping definition for each entity type and link type.

The i2 Analyze deployment toolkit and the ETL toolkit both have a command for adding information about an ingestion source to the Information Store.

The deployment toolkit command looks like this:

```
setup -t addInformationStoreIngestionSource
      -p ingestionSourceName=src_name
      -p ingestionSourceDescription=src_display_name
```

While the ETL toolkit command looks like this:

```
addInformationStoreIngestionSource
      -n src_name
      -d src_display_name
```

In both cases, `src_name` is a unique name for the ingestion source, which also appears in the mapping file. `src_display_name` is a friendlier name for the ingestion source that might appear in the user interface of applications that display records from the Information Store.

Important: The value that you provide for `src_name` must be 30 characters or fewer in length. Also, do not use the word ANALYST as the name of your ingestion source. That name is reserved for records that analysts create in the Information Store through a user interface.

1. If you are using the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory. If you are using the ETL toolkit, navigate to the `etltoolkit` directory.
2. Run the `addInformationStoreIngestionSource` command, specifying the short and display names of your ingestion source.

For example:

```
setup -t addInformationStoreIngestionSource
      -p ingestionSourceName=EXAMPLE
      -p ingestionSourceDescription="Example data source"
```

If the Information Store already contains information about an ingestion source with the name EXAMPLE, this command has no effect.

After you complete this task, you have performed all the necessary actions, and gathered all the necessary information, to be able to write ingestion mapping files. The next task is to create the ingestion mapping file for your ingestion source.

Creating an ingestion mapping file

The mappings in an ingestion mapping file define how rows in staging tables become i2 Analyze records in the Information Store during the ingestion process. Each mapping that you write describes how to construct the origin identifiers for data of a particular type, and specifies the security dimension values that apply to records.

Before you create your ingestion mappings, review the information about [origin identifiers](#) and [security dimension values](#) in i2 Analyze.

The Information Store ingestion mechanism makes it possible for you to develop and extend your ingestion mappings over time. You can test your approach to ingestion by writing and using a single (entity type) mapping, and then adding more entity type and link type mappings later. You can put all your mappings in one file, or put each mapping in a separate file, or anything between those two extremes.

If you populated the staging tables successfully, then writing ingestion mappings can be straightforward. Eventually, you need a mapping for each staging table that you created, but you can approach the problem one mapping at a time.

1. Choose a populated staging table for an entity type that has links to other entity records in the data model.
2. Create an [ingestion mapping file](#) that contains an ingestion mapping for the staging table that you chose in step 1.

If you prefer to start from an existing file, look at `mapping.xml` in the `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit.

3. Run the ingestion command to [validate](#) the mapping.

If you are unhappy with the outcome, edit the ingestion mapping and run the command again.

4. Repeat all of the preceding steps for all the other staging tables that you populated.

The `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit contains an ingestion mapping file named `mapping.xml`. This file contains ingestion mappings for all the staging tables that the ingestion example creates. You can use `mapping.xml` as the basis for the ingestion mappings that you need for your data.

Validating the ingestion configuration

After you create and populate your staging tables and write your ingestion mappings, the final part of the process is to run the command to validate your work.

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for validating your staging tables and a particular ingestion mapping in a particular mapping file.

The procedure for instructing the Information Store to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. Choose an entity staging table that you populated with data and provided with an ingestion mapping.
2. Run the **ingestInformationStoreRecords** command in VALIDATE mode.

For example:

```
setup -t ingestInformationStoreRecords -p importMappingsFile=mapping.xml  
-p importMappingId=Person -p importMode=VALIDATE
```

When you specify `importMode` and set it to `VALIDATE`, the command checks the validity of the specified mapping, but no ingestion takes place. For more information about the running the command and any arguments, see [“The ingestInformationStoreRecords task”](#) on page 406.

The output to the console indicates whether the mapping you identified is valid, provides guidance when it is not valid, and gives a full list of column mappings.

The command sends the same information to a log file that you can find at `toolkit\configuration\logs\importer\IBM_i2_Importer.log`.

3. Complete this validation step for each staging table and ingestion mapping that you plan to use.

The ingestion process for links verifies that the entities at each end of the link are already ingested. If it fails to find them, the process fails. When you are developing your ingestion process, ingest a small amount of entity data before you validate your links.

Correct any problems in the ingestion mappings file (or any ingestion properties file that you specified) before you proceed to [“Adding data to the Information Store”](#) on page 378.

Adding data to the Information Store

After you populate the staging tables and write ingestion mappings, you can use toolkit commands to instruct the Information Store to ingest or update the records that represent external data. The Information Store keeps a log of all such instructions that you can review to determine the success or failure of each one.

The commands in the i2 Analyze deployment and ETL toolkits make it possible for you to create, update, and delete records in the Information Store. All three operation types are controlled by the data in the staging tables and the mappings in the ingestion mapping files.

After any operation that uses toolkit commands to change the contents of the Information Store, you can examine ingestion reports to determine how successful the operation was.

As described in [“Information Store data ingestion”](#) on page 367, there are 2 different import modes that you can use to ingest your data. Before you run the ingestion commands, ensure that you use the correct import mode for the data that you want to ingest. Remember that this might differ depending on the item type that you are ingesting or ingestion mapping that you are using.

Using bulk import mode

Bulk import mode can be used for improved ingestion performance when you are populating the Information Store with new records as part of an initial or periodic load.

- Before you run the ingestion commands, ensure that you complete all of the tasks in [“Preparing for ingestion”](#) on page 370.
- For Db2, the mechanism that bulk import mode uses to load the data from the staging tables to the Information Store can require changes to the database configuration. Before you ingest data, ensure that your database is configured correctly. For more information, see [“Database configuration for IBM Db2”](#) on page 380.
- You can drop and re-create the database indexes during the ingestion process, which can reduce the total time that it takes to ingest data. For more information about when it is beneficial to drop and re-create the database indexes, see [“Database index management”](#) on page 380.

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for ingesting the data that is associated with a particular ingestion mapping in a particular mapping file. For more information about the running the command and any arguments, see [“The ingestInformationStoreRecords task”](#) on page 406.

When you use bulk import mode, take the following points into consideration:

- For best performance, ensure that there are no analysts using the system during the ingestion process.
- Do not include correlation identifiers in the data that you ingest by using bulk import mode. If any correlation identifiers are included, the ingestion fails.
- You cannot run concurrent `ingestInformationStoreRecords` commands when using bulk import mode.

After you stage your data and create your configuration files, you can instruct the Information Store to ingest your data by using bulk import mode. The procedure to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. If you have decided to drop the database indexes, do so now.
2. Use the following command to ingest your data.

```
setup -t ingestInformationStoreRecords
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p importMode=BULK
```

For more information about the running the command and any arguments, see [“The ingestInformationStoreRecords task”](#) on page 406.

3. If any errors occur, refer to [“Troubleshooting the ingestion process”](#) on page 412.
4. If you need to re-create the database indexes, do so now.

Database configuration for IBM Db2

When you use bulk or bulk delete import modes, the way that the data is inserted into or removed from the Information Store from the staging table requires more transaction log space and database locks available than the standard modes.

Transaction log size

For entity operations, allocate at least 1.5 GB of transaction log space.

For link operations, allocate at least 22 GB of transaction log space.

For more information about changing the Db2 log file size, see [logfilsiz - Size of log files configuration parameter](#), [logprimary - Number of primary log files configuration parameter](#), and [logsecond - Number of secondary log files configuration parameter](#).

Lock list size

For entity operations, the formula for calculating the required lock list size is:

```
1200000 * (lock_record_size / 4000)
```

If the lock record size for your deployment is 256 bytes, you need a lock list size of 75,000.

For link operations, the formula for calculating the required lock list size is:

```
3600000 * (lock_record_size / 4000)
```

If the lock record size for your deployment is 256 bytes, you need a lock list size of 225,000. Link ingestion requires an increased lock list size because more tables in the Information Store are written to.

For information about lock lists, see [locklist - Maximum storage for lock list configuration parameter](#).

Database index management

It can be beneficial for performance to drop and re-create indexes in the Information Store database during bulk mode ingestion. The situations when dropping and creating the database indexes might help change over the lifetime of a deployment as your Information Store contains more data.

Decide when to drop and create indexes

The situations where you might want to drop and re-create the database indexes include:

- Completing an initial load for an item type or ingesting data into an empty Information Store
- When ingesting large amounts of data (For example, more than 20 million rows in the staging table)

The situation where you do not want to drop and re-create the database indexes include:

- When the Information Store contains a large amount of data for the item type you are ingesting, the time that it takes to re-create the indexes can make the total ingestion time longer. If you are ingesting into an Information Store that already contains about 1 billion records, do not drop and re-create the database indexes.
- If you are ingesting links between entities of the same type and some of those entities might be correlated, do not drop and re-create the database indexes.

By default, the indexes are kept in place and not dropped during data ingestion.

To help determine what is best for your data, you can test some ingestions that drop and re-create indexes and some that don't. When you complete the test ingestions, record the time that it takes to complete. You can use these times to inform you whether to manage the database indexes during ingestion or not. As the amount of data in the Information Store increases, the time it takes to create the indexes also increases.

Dropping and re-creating indexes during a bulk import ingestion process

If you decide to drop and re-create the database indexes during large ingestions that use multiple staging tables across both entity and link types, the high-level process consists of the following steps:

1. Stop Liberty and Solr
2. Drop the database indexes for the entity types that you are ingesting data for
3. Ingest all entity data
4. Create the database indexes for the entity types that you ingested data for
5. Drop the database indexes for the link types that you are ingesting data for
6. Ingest all link data
7. Create the database indexes for the link types that you ingested data for
8. Start Liberty and Solr

You can use two methods to drop and create indexes:

- Use the i2 Analyze deployment toolkit to generate scripts that you run against the database manually.
- Use the import configuration properties file to specify whether the indexes must be dropped or created during a bulk import mode ingestion.

If you are ingesting data for multiple link types or you want to review the database scripts and run them manually, it is best to use the generated scripts to drop and re-create the database indexes. If you do not want to, or cannot, run scripts against the Information Store database, you can use the import configuration file and allow the ingestion process to drop and re-create the database indexes.

If you are completing an ingestion that spans multiple ingestion periods where the system is in use between ingestion periods, ensure that all of the indexes are created at the end of an ingestion period and that you start Liberty and Solr before analysts use the system.

For information about creating the import configuration file, see [References and system properties](#).

Using the generated scripts method

1. In your import configuration file, set both the `dropIndexes` and `createIndexes` settings to `FALSE`.

For example:

```
dropIndexes=FALSE
createIndexes=FALSE
```

2. Generate the *create* and *drop* index scripts for each item type that you are ingesting data for.

For example, to generate the scripts for the item type with identifier ET5, run the following commands:

```
setup -t generateInformationStoreIndexCreationScripts -p schemaTypeId=ET5
setup -t generateInformationStoreIndexDropScripts -p schemaTypeId=ET5
```

The scripts are output in the `toolkit\scripts\database\db2\InfoStore\generated` directory, in the `createIndexes` and `dropIndexes` directories.

3. Stop Liberty and Solr.

On the Liberty server, run:

```
setup -t stopLiberty
```

On each Solr server, run:

```
setup -t stopSolrNodes --hostname solr.host-name
```

Where *solr.host-name* is the host name of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

4. Run the scripts that you generated in step 2 to drop the indexes for each entity type that you plan to ingest data for. For example:

```
db2 -tvf ET5-drop-indexes.db2
```

Where *ET5* is the item type identifier.

5. Complete the process to ingest the entity data using the bulk import mode. For more information, see [“Using bulk import mode”](#) on page 379.

6. Run the scripts that you generated in step 2 to create the indexes for each entity type that you ingested. For example:

```
db2 -tvf ET5-create-indexes.db2
```

7. Run the scripts that you generated in step 2 to drop the indexes for each link type that you plan to ingest data for. For example:

```
db2 -tvf LT1-drop-indexes.db2
```

Where *LT1* is the item type identifier.

8. Complete the process to ingest the link data using the bulk import mode. For more information, see [“Using bulk import mode”](#) on page 379.

Only re-create the database indexes after you ingest the link data for every link type that you plan to ingest.

9. Run the scripts that you generated in step 2 to create the indexes for each link type that you ingested. For example:

```
db2 -tvf LT1-create-indexes.db2
```

10. Start Liberty and Solr.

On each Solr server, run:

```
setup -t startSolrNodes --hostname solr.host-name
```

Where *solr.host-name* is the host name of the Solr server where you are running the command, and matches the value for the *host-name* attribute of a `<solr-node>` element in the `topology.xml` file.

On the Liberty server, run:

```
setup -t startLiberty
```

Using the import configuration properties file method

When you use the import configuration settings to drop and re-create the database indexes, the toolkit creates and drops the indexes for you as part of the ingestion process instead of running scripts against the database manually. However, you must modify the import configuration file a number of times throughout the ingestion of multiple item types. You must still stop Liberty and Solr before you run the ingestion command, and start them again after the indexes are created.

If all of the data for a single item type is ingested with a single ingestion command, in your import configuration file set the `dropIndexes` and `createIndexes` settings as follows:

```
dropIndexes=TRUE  
createIndexes=TRUE
```

If the data for a single item type must be ingested by using multiple ingestion commands, you need to modify the import configuration file before the first ingestion command, for the intermediate commands, and before the final command for each item type. For example, if your data for a single entity type is in more than one staging table.

1. The first time you call the ingestion command, set the `dropIndexes` and `createIndexes` settings as follows:

```
dropIndexes=TRUE  
createIndexes=FALSE
```

2. For the intermediate times that you call the ingestion command, set the `dropIndexes` and `createIndexes` settings as follows:

```
dropIndexes=FALSE  
createIndexes=FALSE
```

3. The final time you call the ingestion command, set the `dropIndexes` and `createIndexes` settings as follows:

```
dropIndexes=FALSE  
createIndexes=TRUE
```

After you ingest all the entity data, repeat the process for the link data.

Using standard import mode

After you create and populate your staging tables, write your ingestion mappings, and validate your work, instruct the Information Store to ingest your data.

If you are using correlation, you must ensure that the application server is started before you run the ingestion commands.

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for ingesting the data that is associated with a particular ingestion mapping in a particular mapping file. For more information about the running the command and any arguments, see [“The ingestInformationStoreRecords task” on page 406](#).

The procedure for instructing the Information Store to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. Choose an entity staging table that you populated with data and provided with an ingestion mapping.
2. Run the **ingestInformationStoreRecords** command in STANDARD mode to instruct the Information Store to ingest your data.

For example:

```
setup -t ingestInformationStoreRecords
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p importMode=STANDARD
```

For more information about the running the command and any arguments, see [“The ingestInformationStoreRecords task” on page 406](#).

Note:

- You can improve the performance of entity ingestion by running **ingestInformationStoreRecords** for different entity types at the same time when you use the standard import mode. It is recommended that you use the ETL toolkit to run concurrent **ingestInformationStoreRecords** commands. *Do not* attempt to run the command for data of the same type at the same time.
 - Due to the increased number of operations and comparisons that are required, using correlation can make the ingestion process take longer.
3. Repeat steps 1 and 2 for the other staging table and ingestion mapping combinations that you created. Take care to run the command for entity types before you run it for link types.

The ingestion process for links verifies that the entities at each end of the link are already ingested. If it fails to find them, the process fails.

At the end of this procedure, all the external data that you populated the staging tables with is in the Information Store. To add or update records, you can repopulate the staging tables and rerun the **ingestInformationStoreRecords** command.

Updating the Information Store for changed data

The data that the Information Store ingests is fixed at the moment of ingestion, and changes to the data in its source do not automatically update the Information Store. However, you can update the Information Store to reflect changes in an external source by running through the ingestion process again.

For most changes to the data in an external source, it is likely that you can reuse the work that you did to enable initial ingestion. If the changes to an external source are not significant enough to affect your method for generating reproducible origin identifiers, repeat ingestion follows the same process as initial ingestion.

1. Examine the new or changed data in the external source, and your ingestion mappings. Confirm that your configuration still generates origin identifiers that the Information Store can compare with their equivalents in existing ingested data.
2. Delete the contents of each staging table that you know to be affected by changes to the external data.
3. Populate the affected staging tables with the latest data from your external source.
4. Run the ingestion command specifying the standard import mode for each ingestion mapping that refers to an affected staging table, taking care to process entity data before link data, as usual.

The Information Store uses the origin identifier of each row that it attempts to ingest to determine whether the data is new:

- If the origin identifier does not match the origin identifier of any data that is already in the Information Store, then the data is new to the Information Store. It is ingested in the usual way.
- If the origin identifier does match the origin identifier of any data that is already in the Information Store, then the staging table contains updated information. The Information Store clears its existing data and refills it with the new data.

Note: If the correlation identifier changed, additional merge and unmerge operations might occur.

After you follow this procedure, the Information Store contains new data that was added to an external source since the last ingestion. It also contains updated data that was changed in an external source since the last ingestion.

Updating the Information Store for deleted data

The data that the Information Store ingests is fixed at the moment of ingestion, and removal of the data in the external source does not automatically delete it from the Information Store. However, you can update the Information Store to reflect the deletion of data in the external source by using staging tables and the deployment toolkit.

When data changes in its original source, you can use the same pipeline that you used for initial ingestion to update the records in the Information Store. If data is deleted from its source, you can use the staging tables and the deployment toolkit to reflect that fact in the Information Store as well.

A single i2 Analyze record can represent data from multiple sources, which results in a record that contains multiple pieces of provenance. As a consequence, responding to source data deletion does not necessarily mean deleting records from the Information Store. When you use the toolkit to reflect deleted source data, the effect is to remove the provenance associated with that data. If the process removes a record's only provenance, the record is deleted. If not, the record remains in the Information Store.

Note: To delete records from the Information Store explicitly, use the deletion-by-rule approach. You can write conditions to determine which records are deleted. For more information about deleting records in this way, see [Deleting records by rule](#).

The commands to update the Information Store for deleted data use the same mapping file and staging tables as the commands for ingesting data, and you call them in a similar way. However, the only information that *must* be in the staging table is what the mapping file requires to generate the origin identifiers of the data that is no longer in the external source.

When you run the commands to update the Information Store for deleted data, the rules that apply differ from the rules for adding and updating data:

- Links do not have to be processed before entities, or vice versa.
- Links can be processed without specifying the origin identifiers of their ends.
- Deleting a piece of provenance from an entity record also deletes all the link provenance that is connected to it.

- The process silently ignores any origin identifiers that are not in the Information Store.

Because this process might cause significant numbers of i2 Analyze records to be deleted, two commands are provided. The first command previews the effect of running the second command before you commit to doing so. In the deployment toolkit, the two commands have different names but the same syntax:

```
setup -t previewDeleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
```

```
setup -t deleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p logConnectedLinks
      -p importMode=BULK_DELETE
```

In the ETL toolkit, you reuse the `ingestInformationStoreRecords` command. For more information about running the command from the ETL toolkit, see [“ETL toolkit”](#) on page 407.

For more information about the running the commands and any arguments, see [“The previewDeleteProvenance and deleteProvenance tasks”](#) on page 408.

Bulk delete mode can be used for improved performance when you are removing provenance from the Information Store that does not contribute to correlated records. If you try to delete any provenance that contributes to correlated records, that provenance is not removed from the Information Store and is recorded in a table in the `IS_Public` database schema. The table name is displayed in the console when the delete process finishes. For example, `IS_Public.D22200707130930400326011ET5`. Before you use bulk delete mode, ensure that your database is configured correctly. For more information, see [“Database configuration for IBM Db2”](#) on page 380.

The procedure for updating the Information Store in this way starts with a staging table that contains information about the data that you no longer want to represent in the Information Store.

1. Ensure that the application server that hosts i2 Analyze is running.
2. Run the `previewDeleteProvenance` command to discover what the effect of running `deleteProvenance` is.

For example:

```
setup -t previewDeleteProvenance -p importMappingsFile=mapping.xml
      -p importMappingId=Person
```

The output to the console window describes the outcome of a delete operation with these settings. High counts or a long list of types might indicate that the operation is going to delete more

records than you expected. Previewing the delete operation does not create an entry in the Ingestion_Deletion_Reports view, the output is displayed in the console.

```
>INFO [DeleteLogger] - Delete preview requested at 2017.12.08 11:05:32
>INFO [DeleteLogger] - Item type: Person
>INFO [DeleteLogger] - Number of 'Person' provenance pieces to be deleted: 324
>INFO [DeleteLogger] - Number of 'Person' i2 Analyze records to be deleted: 320
>INFO [DeleteLogger] - Number of 'Access To' provenance pieces to be deleted: 187
>INFO [DeleteLogger] - Number of 'Access To' i2 Analyze records to be deleted: 187
>INFO [DeleteLogger] - Number of 'Associate' provenance pieces to be deleted: 27
>INFO [DeleteLogger] - Number of 'Associate' i2 Analyze records to be deleted: 27
>INFO [DeleteLogger] - Number of 'Employment' provenance pieces to be deleted: 54
>INFO [DeleteLogger] - Number of 'Employment' i2 Analyze records to be deleted: 54
>INFO [DeleteLogger] - Number of 'Involved In' provenance pieces to be deleted: 33
>INFO [DeleteLogger] - Number of 'Involved In' i2 Analyze records to be deleted: 33
>INFO [DeleteLogger] - Duration: 1 s
```

Note: When you run the command for entity records, the output can exaggerate the impact of the operation. If the staging table identifies the entities at both ends of a link, the preview might count the link record twice in its report.

3. Correct any reported problems, and verify that the statistics are in line with your expectations for the operation. If they are not, change the contents of the staging table, and run the preview command again.
4. Run the `deleteProvenance` command with the same parameters to update the Information Store. For example:

```
setup -t deleteProvenance -p importMappingsFile=mapping.xml
      -p importMappingId=Person -p importLabel=DeletePeople
      -p logConnectedLinks
```

Note: Do not run multiple `deleteProvenance` commands at the same time, or while data is being ingested into the Information Store.

5. Repeat the steps for the types of any other records that you want to process.

At the end of this procedure, the Information Store no longer contains the provenance (or any connected link provenance) for the data that you identified through the mapping files and staging tables. Any records that lose all of their provenance, and any connected link records, are deleted as a result. Deleting data is permanent, and the only way to restore it to the Information Store is to add it again through the `ingestInformationStoreRecords` command.

Ingestion resources

The ingestion resources section contains information that is referenced elsewhere in the ingestion section.

Understanding the architecture

The physical architecture of your i2 Analyze deployment both affects and is affected by how you acquire and transform external data for the Information Store. Depending on the architecture, you might need to perform more deployment tasks before you can run data ingestion commands.

A complete solution for loading and ingesting data into the Information Store has four mandatory architectural components:

- The i2 Analyze server, and in particular the deployment toolkit that it contains

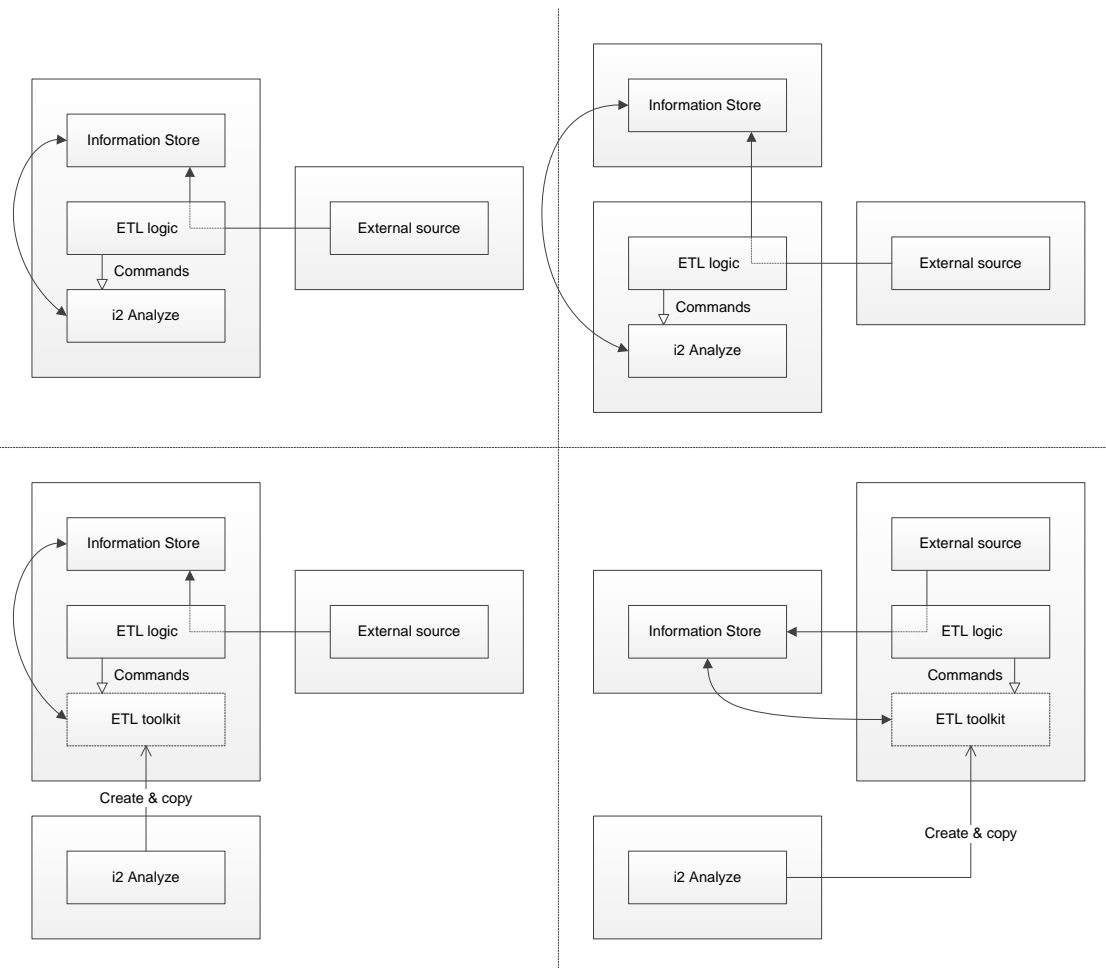
- The database management system that contains the Information Store and the staging tables
- An external data source
- The ETL logic that transforms the source data and loads it into the staging tables

Note: In addition to the mandatory components, you can opt to include a component that prepares your data for correlation, such as a matching engine or a context computing platform. These components can help you to determine when data in multiple different sources represents the same real-world "thing". This preparation of your data might occur as part of your ETL logic, or between an external data source and the ETL logic.

i2 Analyze supports physical architectures in which the database is hosted on the same server as the application, or on a different one. You can also choose to locate your ETL logic on the same server as the i2 Analyze application, or on the same server as the database, or on an entirely separate server.

The process of transforming source data can be demanding, especially if the requirements are complex or the volume is high. There are also scenarios in which you might want to automate the process of loading and then ingesting the external data. Ultimately, the architecture that you decide upon depends on the needs and constraints of your deployment.

The following diagram shows some of the permutations. The examples in the upper-left and upper-right quadrants represent deployments in which the ETL logic (implemented by a tool like IBM DataStage, for example) is co-hosted with the i2 Analyze application. The database can be on the same or a separate server; the solid arrows show data flow between the components during data load and ingestion.



The examples in the lower-left and lower-right quadrants represent deployments in which the ETL logic is on a separate server from the i2 Analyze application. (Typically, the ETL logic is hosted alongside the database or the external data source.) To enable the architecture, those deployments include the *ETL toolkit*, which is a cut-down version of the main deployment toolkit that targets only data ingestion.

When you need the ETL toolkit, you can generate it on the i2 Analyze server, and copy it to the server that hosts the ETL logic. When the ETL toolkit is properly configured, your ETL logic can run toolkit commands without reference to the rest of the deployment. If you decide to use the ETL toolkit, the next step is to [deploy it](#). If not, you can move on to [creating staging tables](#) in the database.

As the diagrams show, the ETL toolkit is most likely to be useful in deployments where the i2 Analyze application and the ETL logic are on separate servers. As you plan your approach to ingestion, consider the following:

- If the ETL logic is relatively simple and data volumes are low, there are benefits to colocating as many components as you can, especially in a new deployment.
- If your deployment requires separate servers from the start, or as it evolves over time, determine where the bottlenecks are. Is it limited by server speed or network speed?

- If the ETL logic is taxing a server that hosts other components, consider moving the logic, but be aware of the increase in network traffic.
- If the volume of data is taxing the network, consider colocating components when you are able. (You might not have permission to deploy components to some servers, for example.)

By acting as a proxy for the i2 Analyze deployment toolkit, the ETL toolkit provides for more flexibility in your choice of architecture. In some circumstances you can separate the database, the ETL logic, and the i2 Analyze application without incurring a networking penalty.

Deploying the ETL toolkit

If your deployment includes logic that extracts, transforms, and loads data on a different server from the i2 Analyze application or the Information Store, consider deploying the ETL toolkit. The ETL logic can then run ETL toolkit commands to automate loading and ingesting data into the Information Store.

In an i2 Analyze deployment that uses data from an external source, the ETL logic is the processing that transforms source data for loading into the Information Store staging tables. In mature deployments, it is common for the ETL process to be automated so that loading and ingesting data happen in sequence, on a schedule.

When your ETL logic is colocated with the standard i2 Analyze deployment toolkit, the logic can use that toolkit to drive the ingestion process automatically. When those components are on separate servers, you can deploy the ETL toolkit to the server that hosts the ETL logic. The ETL toolkit provides the ingestion functions of the deployment toolkit in a stand-alone package.

The ETL toolkit must be able to communicate with the Information Store with all the same credentials as the deployment toolkit. To enable this behavior, you use the deployment toolkit to create the ETL toolkit, and then copy it to the ETL logic server.

1. On the server that has the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory.
2. Run the `createEt1Toolkit` command to generate the ETL toolkit:

```
setup -t createEt1Toolkit -p outputPath=output_path
```

This command creates the ETL toolkit in a directory that is named `et1toolkit` in the output path that you specify.

3. Copy the ETL toolkit to the server that hosts the ETL logic.

If the ETL logic and toolkit are on the same server as the database management system that hosts the Information Store, you do not need to modify the connection configuration. If the database management system is on a different server, then you must ensure that the ETL toolkit can communicate with the remote database.

4. Dependent on your database management system, install either Db2 client software or Microsoft Command Line Utilities for SQL Server on the server that hosts the ETL toolkit.
For more information, see [Software Prerequisites](#).
5. Navigate to the `classes` directory of the ETL toolkit and open the `Connection.properties` file in a text editor.
6. Ensure that the value for the `db.installation.dir` setting is correct for the path to the Db2 client or Microsoft Command Line Utilities for SQL Server on the server that hosts this ETL toolkit.
For example:

```
db.installation.dir=C:/Program Files/IBM/SQLLIB
```

7. If you are using Db2 to host the Information Store, you must catalog the remote Db2 database. Run the following commands to enable the ETL toolkit to communicate with the Information Store:

```
db2 catalog tcpip node node-name host-name server port-number
db2 catalog database instance-name at node node-name
```

Here, *host-name*, *port-number*, and *instance-name* are the values that are specified in the `topology.xml` file. *node-name* can be any value that you choose, but you must use the same value in both commands.

If the database management system that hosts the Information Store is not using SSL, then the process is complete.

If the database management system is configured to use SSL, you must also enable the ETL toolkit to communicate by using SSL.

8. Register the `i2-database_management_system-certificate.cer` certificate that you exported from the database management system when you configured SSL on the server that hosts the ETL toolkit.
 - On Windows, import the certificate into the Trusted Root Certification Authorities store for the current user.
 - On Linux, copy the certificate to the `/etc/pki/ca-trust/source/anchors` directory and use `update-ca-trust` to enable it as a system CA certificate.
9. Create a truststore and import into the truststore the certificate that you exported from the database management system when you configured SSL.

For example, run the following command:

```
keytool -importcert -alias "dbKey"
-file C:\IBM\etltoolkit\i2-database_management_system-certificate.cer
-keystore "C:\IBM\etltoolkit\i2-etl-truststore.jks"
-storepass "password"
```

Enter yes in response to the query, Trust this certificate?

10. Navigate to the `classes` directory of the ETL toolkit and open the `TrustStore.properties` file in a text editor.
 11. Populate the `DBTrustStoreLocation` and `DBTrustStorePassword` properties with the full path to the truststore that you created, and the password that is required to access it.
- For example:

```
DBTrustStoreLocation=C:/IBM/etltoolkit/i2-etl-truststore.jks
DBTrustStorePassword=password
```

12. You can use the Liberty profile `securityUtility` command to encode the password for the truststore.
 - a) Navigate to the `bin` directory of the WebSphere Application Server Liberty profile deployment that was configured by the deployment toolkit.
 - b) In a command prompt, run `securityUtility encode password`, which generates and displays the encoded password.

Use the entire value, including the `{xor}` prefix, for the `DBTrustStorePassword` property value. For more information about using the security utility, see [securityUtility command](#).

The ETL toolkit is ready for use by your ETL logic to modify the Information Store. At key points in the processes of preparing for and performing ingestion, you can use commands in the ETL toolkit in place of deployment toolkit functions.

Information Store staging tables

At your request, i2 Analyze generates an Information Store staging table that can contain data for i2 Analyze records of a single entity type or link type. To generate the staging table, it uses information from the i2 Analyze schema, which is the same starting point from which it generates the main data tables during deployment.

An entity type staging table contains:

- At least one column for each property type in the schema.
- Columns for storing information that uniquely identifies data in its source. During ingestion, i2 Analyze can use this information to construct an origin identifier for the ingested data.
- Two columns to record when the data was created and updated in the source.
- Three columns for storing information that describes the source of the data. During ingestion, i2 Analyze uses this information to populate a source reference for the ingested data.
- Two columns to record the *correlation identifier type* and *correlation identifier key* of the data. During ingestion, i2 Analyze uses the information in these columns to construct the correlation identifier for the ingested data.
- A column for each security dimension that the security schema defines. During ingestion, i2 Analyze can use the information in these columns to implement per-record security.

For example, if the i2 Analyze schema contains this simplified entity type definition:

```
<EntityType Id="ET5" DisplayName="Person">
  <PropertyTypes>
    <PropertyType DisplayName="First (Given) Name"
      LogicalType="SINGLE_LINE_STRING" Id="PER4"/>
    <PropertyType DisplayName="Birth place location"
      LogicalType="GEOSPATIAL" Id="PER5"/>
    <PropertyType DisplayName="Date of Birth"
      LogicalType="DATE" Id="PER9"/>
    <PropertyType DisplayName="Date and Time of Death"
      LogicalType="DATE_AND_TIME" Id="PER10"/>
  </PropertyTypes>
</EntityType>
```

This SQL statement is then the definition of a corresponding staging table in Db2:

```
CREATE TABLE "IS_Staging"."E_Person" (  
    "source_id" VARCHAR(50),  
    "origin_id_type" VARCHAR(100),  
    "origin_id_keys" VARCHAR(1000),  
    "source_created" TIMESTAMP,  
    "source_last_updated" TIMESTAMP,  
    "correlation_id_type" VARCHAR(100),  
    "correlation_id_key" VARCHAR(1000),  
    "p_first_given_name" VARCHAR(250),  
    "p_birth_place_location" VARCHAR(250),  
    "p_date_of_birth" DATE,  
    "p0_date_and_time_of_deat" TIMESTAMP,  
    "p1_date_and_time_of_deat" VARCHAR(250),  
    "p2_date_and_time_of_deat" SMALLINT,  
    "p3_date_and_time_of_deat" TIMESTAMP,  
    "security_level" VARCHAR(50),  
    "security_compartment" VARCHAR(50),  
    "source_ref_source_type" VARGRAPHIC(200),  
    "source_ref_source_location" DBCLOB(2000),  
    "source_ref_source_image_url" DBCLOB(2000)  
);
```

And this SQL statement is then the definition of a corresponding staging table in SQL Server:

```
CREATE TABLE IS_Staging.E_Person(  
    source_id nvarchar(50) NULL,  
    origin_id_type nvarchar(100),  
    origin_id_keys nvarchar(1000),  
    source_created datetime2(6) NULL,  
    source_last_updated datetime2(6) NULL,  
    correlation_id_type nvarchar(100) NULL,  
    correlation_id_key nvarchar(1000) NULL,  
    p_first_given_name varchar(250) NULL,  
    p_birth_place_location varchar(max) NULL,  
    p0_date_and_time_of_deat datetime2(4) NULL,  
    p1_date_and_time_of_deat nvarchar(250) NULL,  
    p2_date_and_time_of_deat smallint NULL,  
    p3_date_and_time_of_deat datetime2(6) NULL,  
    security_level nvarchar(50) NULL,  
    security_compartment nvarchar(50) NULL,  
    source_ref_source_type nvarchar(200) NULL,  
    source_ref_source_location nvarchar(2000) NULL,  
    source_ref_source_image_url nvarchar(2000) NULL  
);
```

Note: Additionally, staging tables for link types contain a column for the direction of the link, and further columns for the information that uniquely identifies the link end data in the source.

The statements create the staging table in a separate schema from the Information Store data tables. Many of the columns in the staging table have names that are derived from the display names of the

property types in the i2 Analyze schema. In most cases, the relationship between the schema and the staging table is obvious, but there are a number of extra columns and differences:

- The `source_id`, `origin_id_type`, `origin_id_keys` columns of the staging table can be used to store values that reference the rest of the data in its original source and can be used to make up the origin identifier of the resulting record.

Note: If the staging table definition was for a link type, it would also contain `from_` and `to_` variations of each of the columns.

For more information about generating origin identifiers during ingestion, see [“Origin identifiers” on page 403](#).

- The next two columns of the staging table are `source_created` and `source_last_updated`. You can use these columns to store information about when the data to be ingested was created and modified in its source.
- The next two columns of the staging table are `correlation_id_type` and `correlation_id_key`. If you want to correlate data during ingestion into the Information Store, you can use these columns to store values that i2 Analyze uses to generate correlation identifiers. For more information, see [Overview of correlation](#).

Note: Although populating the correlation identifier columns is not mandatory, doing so acts like a switch. The presence of correlation identifier values in any row of a staging table causes i2 Analyze to perform correlation for all the rows in that table.

- Any property type in the i2 Analyze schema that has the logical type `DATE_AND_TIME` occupies four columns in the staging table. These columns always appear in the same order:

- The "P0" column is for the local date and time as originally recorded, as a `DATE_AND_TIME`.
- The "P1" column is for the time zone of the local date and time, as listed in the [IANA database](#). For example, Europe/London.
- The "P2" column is for an indicator of whether Daylight Saving Time is (1) or is not (0) in effect.

Note: i2 Analyze considers this value only when the time is ambiguous because it occurs during the hour that is "repeated" when Daylight Saving Time ends.

- The "P3" column is for the date and time as expressed in Coordinated Universal Time (UTC), as another `DATE_AND_TIME`.

For more information about the permitted values for `DATE_AND_TIME` columns in your database management system, see [“Information Store property value ranges” on page 395](#).

- The next columns derive from the security schema rather than the i2 Analyze schema. One column exists for each security dimension that the security schema defines. You can use these columns if you want to give different dimension values to each i2 Analyze record that is created or updated as a result of ingestion.
- In link tables, there is also a `direction` column to store the direction of links.
- The final three columns are named `source_ref_source_type`, `source_ref_source_location`, and `source_ref_source_image_url`. These columns are used to populate the source reference that is generated when the data is ingested.

For more information about implementing source references in your system, see [Configuring source references](#).

The staging tables contain some, but never all, of the data for i2 Analyze records. They do not contain the type identifiers that Information Store records must have, and it is not mandatory to populate the columns for timestamps, security dimension values, or correlation identifiers. You can supply the remainder of the information in an ingestion mapping.

Information Store property value ranges

The Information Store places limits on the ranges of values that properties can contain. Different logical types in the i2 Analyze schema imply different limits, which are not always the same as the restrictions on the underlying database. It is important to consider the limits when you prepare data for ingestion.

Logical type	Db2 values	SQL Server values
SINGLE_LINE_STRING	Up to 8000 bytes of UTF-8 characters. The default value is 250.*	Up to 4000 bytes of UTF-16 characters. The default value is 250.*
MULTI_LINE_STRING	Up to 32700 bytes of UTF-8 characters	Up to $2^{31}-1$ bytes of UTF-16 characters.
SELECTED_FROM	Same as SINGLE_LINE_STRING	Same as SINGLE_LINE_STRING
SUGGESTED_FROM	Same as SINGLE_LINE_STRING	Same as SINGLE_LINE_STRING
DATE	From 1753-01-01 to 9999-12-31	From 1753-01-01 to 9999-12-31
TIME	From 00:00:00 to 23:59:59 **	From 00:00:00 to 23:59:59
DATE_AND_TIME	From 1753-01-01T00:00:00Z to 9999-12-31T23:59:59Z **	From 0001-01-01 00:00:00.00Z to 9999-12-31 23:59:59.99Z
BOOLEAN	true or false	true or false
INTEGER	From -2^{31} to $2^{31} - 1$	From -2^{31} to $2^{31} - 1$
DOUBLE	From 4.9×10^{-324} to $1.79769313486231 \times 10^{308}$ (Equivalent range for negative values. Maximum 15 digits of precision.)	From $-1.79E+308$ to $-2.23E-308$, 0 and $2.23E-308$ to $1.79E+308$ (Maximum 17 digits of precision.)
DECIMAL	From -9999999999999999.9999 to 9999999999999999.9999 (Maximum 18 digits before the decimal mark. Maximum 4 digits after it.)	From -9999999999999999.9999 to 9999999999999999.9999 (Maximum 18 digits before the decimal mark. Maximum 4 digits after it.)
GEOSPATIAL	From -180.0000 to 180.0000 for longitude values, and -90.0000 to 90.0000 for latitude values. The data must be in the <u>Well-known text representation</u> , <code>POINT(longitude latitude)</code> . The longitude and latitude values must conform to the WGS84 Bounds. For more information about the values, see <u>Spatial reference - WGS84</u> .	From -180.0000 to 180.0000 for longitude values, and -90.0000 to 90.0000 for latitude values. The data must be in the <u>Well-known text representation</u> , <code>POINT(longitude latitude)</code> . The longitude and latitude values must conform to the WGS84 Bounds. For more information about the values, see <u>Spatial reference - WGS84</u> .

* The value for SINGLE_LINE_STRING depends on the value that is specified for each property with that type in the i2 Analyze schema.

** If a Db2 database underlies the Information Store you can load time values that represent midnight as 24:00:00. When it stores such values, the database converts them to fit the ranges in the table.

Data that includes non-printing or control characters might not be indexed and can cause errors.

In addition to the values in the table, you can set the value of any non-mandatory property to null. In the staging table for an item type that has a DATE_AND_TIME property type, all four columns that the value is spread across must be null in that case.

Ingestion mapping files

An ingestion mapping file is an XML document whose structure is validated during the ingestion process. Every time that you instruct the Information Store to ingest data, you specify both the mapping file to use, and the ingestion mapping within it. You can choose to put all your ingestion mappings in one file, or to spread them across several files.

Ingestion mappings have two complementary purposes. First, they make the association between an entity type or a link type in the i2 Analyze schema and a staging table in the database. Second, they provide any extra information that the Information Store requires but the staging tables do not contain.

For all record types, the extra information that an ingestion mapping can provide includes:

- The type identifier of the entity or link type that the mapping applies to
- The name of the data source that the data to be ingested comes from
- How to create an origin identifier for data of this type
- The security dimension values that all records of this type receive, if you do not use per-record security

Link type ingestion mappings provide further information that addresses the requirements of link records:

- The Information Store must be able to test that it already contains the entity records at the ends of an incoming link. The link type mapping must describe how to create the origin identifiers that are associated with those records so that the Information Store can look them up.
- To make the look-up more efficient, the link type mapping also contains the type identifiers of the entity records that appear at the "from" and "to" ends of the incoming links. A link type that can connect entities of several different types requires a separate mapping for each valid combination of end types.

Ingestion mapping syntax

The root element of an ingestion mapping file is an <ingestionMappings> element from the defined namespace. For example:

```
<ns2:ingestionMappings
  xmlns:ns2="http://www.i2group.com/Schemas/2016-08-12/IngestionMappings"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ...
</ns2:ingestionMappings>
```

Within the ingestion mapping file, you use an <ingestionMapping> element to define a mapping for a particular entity type or link type. Each <ingestionMapping> element has a mandatory id attribute

that must be unique within the mapping file. You use the value to identify the mapping when you start ingestion. For example:

```
<ingestionMapping id="Person">
  ...
</ingestionMapping>
```

Note: For examples of complete ingestion mapping files, search for files with the name `mapping.xml` in the i2 Analyze deployment toolkit. All of those files contain definitions that are similar to the definitions here.

Entity type ingestion mappings

When the mapping is for an entity type, the `<ingestionMapping>` element has the following children:

stagingArea

The `<stagingArea>` element specifies where the mapping gets its staged data from. In this version of i2 Analyze, the staged data is always in a staging table, and `<stagingArea>` always has a `<tableName>` child.

tableName

The value of `<tableName>` is the name of the staging table that contains the data to be ingested.

For example:

```
...
<stagingArea xsi:type="ns2:databaseIngestionSource">
  <tableName>IS_Staging.E_Person</tableName>
</stagingArea>
...
```

itemTypeId

The value of the `<itemTypeId>` element is the identifier of the entity type (or the link type) to which the mapping applies, as defined in the i2 Analyze schema.

For example:

```
...
<itemTypeId>ET5</itemTypeId>
...
```

originId

The `<originId>` element contains a template for creating the origin identifier of each ingested row. `<originId>` has two mandatory child elements: `<type>` and `<keys>`.

For example:

```

...
<originId>
  <type>$(origin_id_type)</type>
  <keys>
    <key>$(origin_id_keys)</key>
  </keys>
</originId>
...

```

Here, `$(origin_id_type)` and `$(origin_id_keys)` are references to the columns named `origin_id_type` and `origin_id_keys` in the staging table to which this ingestion mapping applies. When the Information Store ingests the data, the values from the staging table become the origin identifier in the Information Store.

For more information about generating origin identifiers during ingestion, see [“Origin identifiers”](#) on page 403.

dataSourceName

The value of the `<dataSourceName>` element identifies the data source from which the data in the staging table came. It must match the name of an ingestion source that you provide to the Information Store during the ingestion process.

For example:

```

...
<dataSourceName>EXAMPLE</dataSourceName>
...

```

createdSource and lastUpdatedSource

By default, the ingestion process automatically puts the values from the `source_created` and `source_last_updated` columns of the staging tables into the Information Store. If you want to use the same values for all ingested data, you can override that behavior by including the non-mandatory `<createdSource>` and `<lastUpdatedSource>` elements and specifying values in the date-time string format for your database management system.

For example:

```

...
<createdSource>2002-10-04 09:21:33</createdSource>
<lastUpdatedSource>2002-10-05 09:34:45</lastUpdatedSource>
...

```

securityDimensionValues

Every row that the Information Store ingests must have at least one security dimension value from each dimension in the security schema. The Information Store staging tables contain a column for each access security dimension that the security schema defines.

In your ingestion process, you can use the staging table columns to store dimension values on a per-row basis. Alternatively, you can specify that all the data that the Information Store ingests through the same mapping get the same security dimension values.

In the ingestion mapping file, the `<securityDimensionValues>` element has `<securityDimensionValue>` children. For per-row security, use the value of each `<securityDimensionValue>` element to reference a security dimension column.

For example:

```
...
<securityDimensionValues>
  <securityDimensionValue>$(security_level)</securityDimensionValue>
  <securityDimensionValue>$(security_compartment)</securityDimensionValue>
</securityDimensionValues>
...
```

In the staging table, the referenced columns can contain either a single dimension value, or a comma-separated list of dimension values.

For per-mapping security, set the value of each `<securityDimensionValue>` element to a security dimension value.

For example:

```
...
<securityDimensionValues>
  <securityDimensionValue>HI</securityDimensionValue>
  <securityDimensionValue>UC</securityDimensionValue>
  <securityDimensionValue>OSI</securityDimensionValue>
</securityDimensionValues>
...
```

In either approach, the values that you specify must be present in the i2 Analyze security schema.

Link type ingestion mappings

When the ingestion mapping is for a link type, the `<ingestionMapping>` element has the same children that entity types require, plus the following ones:

fromItemTypeId

The value of the `<fromItemTypeId>` element is the type identifier of entity records that the schema permits at the "from" end of the link type to which this mapping applies.

For example:

```
...
<fromEntityTypeId>ET5</fromEntityTypeId>
...
```

fromOriginId

The `<fromOriginId>` element contains a template for creating the origin identifier of the entity record at the "from" end of each ingested link row. Its syntax is identical to the `<originId>` element.

The origin identifiers that result from `<fromOriginId>` must match the origin identifiers that result from the `<originId>` element for the entity type in question. The ingestion process uses this

information to verify that the Information Store already ingested an entity record that has this origin identifier.

For example:

```
...
<fromOriginId>
  <type>$(from_origin_id_type)</type>
  <keys>
    <key>$(from_origin_id_keys)</key>
  </keys>
</fromOriginId>
...
```

For more information about generating origin identifiers during ingestion, see [“Origin identifiers”](#) on page 403.

toItemId

The value of the `<toItemId>` element is the type identifier of entity records that the schema permits at the "to" end of the link type to which this mapping applies.

For example:

```
...
<toItemId>ET10</toItemId>
...
```

toOriginId

The `<toOriginId>` element behaves identically to the `<fromOriginId>` element, except that it applies to the entity record at the "to" end of each ingested link row.

For example:

```
...
<toOriginId>
  <type>$(to_origin_id_type)</type>
  <keys>
    <key>$(to_origin_id_keys)</key>
  </keys>
</toOriginId>
...
```

For more information about generating origin identifiers during ingestion, see [“Origin identifiers”](#) on page 403.

linkDirection

The `<linkDirection>` element is a non-mandatory child of the `<ingestionMapping>` element. When you include a `<linkDirection>` element in an ingestion mapping, you can either provide the same value for all links, or refer to the direction column of the staging table. Legal values for the element or the column are WITH, AGAINST, BOTH, and NONE.

For example, to use a fixed value:

```
...
<linkDirection>WITH</linkDirection>
...
```

Or, to use the value in the direction column:

```
...
<linkDirection>$(direction)</linkDirection>
...
```

If an ingestion mapping for a link type does not contain a `<linkDirection>` element, then any links that the Information Store ingests through the mapping have no direction.

References and system properties

In an ingestion mapping, you can use constants or references to specify values for i2 Analyze records. When you use a reference, the ingestion process retrieves a value from a staging table column or a property in a settings file. The settings file can also set system properties that control some aspects of ingestion into the Information Store.

By default, the settings file does not exist. You must create the settings file and specify the file when you run the ingestion command.

For example, you might call your settings file `ingestion_settings.properties` and it might contain the following name-value pairs :

```
SEC_LEVEL_VALUE=UC
SEC_COMPARTMENT_VALUE=HI,OSI
IngestionFailureMode=MAPPING
```

When you run the ingestion command, you reference your settings file as follows:

```
setup -t ingestInformationStoreRecords
...
-p importConfigFile=ingestion_settings.properties
```

- [“System properties” on page 401](#)
- [“References” on page 403](#)

System properties

As well as providing values for ingestion mappings, you can use the settings file to configure the behavior of the ingestion process. The file supports a handful of system properties that you can set in the same way as you create and set custom properties.

IngestionFailureMode [RECORD | MAPPING]

When the Information Store encounters a problem with a record during ingestion, its default behavior is to log the error and move on to the next record. Failure is *record-based*. Instead, you can specify that a problem with one record causes the Information Store not to ingest any of the records from that staging table. Failure then is *mapping-based*.

In the settings file, the possible values for the `IngestionFailureMode` setting are `RECORD` or `MAPPING`. The default value is `RECORD`.

For example, to change the failure mode to mapping, add the following line to your settings file:

```
IngestionFailureMode=MAPPING
```

RecordFailureThreshold

During the ingestion process, if the number of errors that occur is greater than the value for the `RecordFailureThreshold` property the process stops and no data is ingested into the Information Store. By default, the value for this property is 1000.

For example:

```
RecordFailureThreshold=500
```

IngestionRunstats [TRUE | FALSE]

If you are using Db2, during the ingestion process the **RUNSTATS** command updates statistics in the system catalog about the characteristics of a table, associated indexes, or statistical views.

In a table that contains many records, it can take the **RUNSTATS** command a long time to complete. If the number of records that you are ingesting is a small percentage of the total number of records in the table, it is recommended that you configure the ingestion process not to call the **RUNSTATS** command. You can then manually run the **RUNSTATS** command after a significant number of records are ingested.

To prevent the ingestion process from calling the **RUNSTATS** command, set the value of `IngestionRunstats` to `FALSE`. For example:

```
IngestionRunstats=FALSE
```

The following command is the **RUNSTATS** command to run manually after an ingestion. The command generates query output, and saves the output to the file named `ISStatisticsCollection.sql` that you can run.

```
db2 connect to ISTORE user <user name> using <password>

db2 -x
"SELECT
    'RUNSTATS ON TABLE ' || TRIM(TABSCHEMA) || '.' || TRIM(TABNAME) || '
    WITH DISTRIBUTION ON ALL COLUMNS
    AND INDEXES ALL
    ALLOW WRITE ACCESS
    TABLESAMPLE BERNOULLI(10)
    INDEXSAMPLE BERNOULLI(10);'
FROM SYSCAT.TABLES
WHERE TYPE = 'T' AND TABSCHEMA = 'IS_DATA'"
> ISStatisticsCollection.SQL
```

ImportBatchSize

The `ImportBatchSize` controls the number of rows from a staging table that are ingested per batch. By default, the `ImportBatchSize` is set to 100,000.

For example:

```
ImportBatchSize=100000
```


If you are ingesting data into the system while analysts are using the system, it is possible that some analysis operations are blocked from returning results while a batch of data is being ingested. During the ingestion process, up to 100,000 rows (or the value of your batch size) can be locked in the database during each batch of the import. These locks cause a potential for the following analytical operations to stop returning results until the batch is complete: Find Path, Expand, and Visual Query.

To determine how long each batch takes to ingest, inspect the `IS_Data.Ingestion_Batches` table. The time that the batch takes to complete is the time that analytical operations might be blocked from returning results. If the time is too long for your requirements, you can reduce the value for `ImportBatchSize` to reduce the batch size and the time that it takes to complete.

For SQL Server, a batch size greater than 1,700 can cause table locks during link ingestion.

References

Many of the pieces of information that you provide in an ingestion mapping are fixed for that mapping. Item types, end types, and some parts of the origin identifier do not change between the i2 Analyze records that one mapping is responsible for. The most appropriate way to specify this kind of information is to use constant values on a per-mapping basis.

The two main reasons for preferring references to constant values lie at opposite ends of the spectrum:

- To give different values for the same field in records that are ingested through the same mapping, you can refer to a staging table column. This approach is appropriate for many non-property values that change from one record to the next.
- To use the same values across multiple ingestion mappings, refer to a property in a settings file. This approach might be appropriate when you want all the data from a source to get the same security dimension values. You can refer to the same property from every mapping that you write.

A settings file that defines properties for the ingestion process is just a text file that contains a set of `name=value` pairs, with one pair on each line:

```
SEC_LEVEL_VALUE=UC
SEC_COMPARTMENT_VALUE=HI,OSI
```

When you run one of the ingestion commands, you can supply it with the name of the properties file whose values you want to use.

To use a value by reference in an ingestion mapping, you use the `$(name)` syntax. `name` is the name of either a column in the staging table or a property in a settings file. For example, `$(SOURCE_ID)` and `$(DIRECTION)` refer to staging table columns, while in the previous example `$(SEC_LEVEL_VALUE)` and `$(SEC_COMPARTMENT_VALUE)` refer to properties.

Note: Since referring to columns and properties uses the same syntax, a clash can happen if a column and a property have the same name. In that case, the value of the property takes precedence.

Origin identifiers

The role of a source identifier is to reference the data for a record reproducibly in its original source. The source identifiers that records receive during ingestion are unique within i2 Analyze, and they have a special name in this context. They are called origin identifiers.

The nature of a origin identifier depends on the source and the creation method, and sometimes on whether the record is a link or an entity. When you ingest data into the Information Store, i2 Analyze compares the incoming origin identifier with existing records. If it finds a match, i2 Analyze updates a record instead of creating one.

After you develop your process for creating origin identifiers, you must continue to use that process. If you change the way that your origin identifiers are created and ingest the same data again, the Information Store creates new records for the data instead of updating the existing records. To ensure that changes to data are processed as updates, you must create your origin identifiers consistently.

For more information about different identifiers in i2 Analyze, see [Identifiers in i2 Analyze records](#).

The structure of an origin identifier

During the ingestion process, you specify the data for your identifiers in the staging table and ingestion mapping file. An origin identifier is constructed of a "type" and "keys".

type

The "type" of an origin identifier allows the services in an i2 Analyze deployment to determine quickly whether they are interested in (or how to process) a particular row of data. The value of the type element does not have to be meaningful, but data from different sources generally have different values.

For a deployment that uses a Db2 database, the length of the origin identifier type must not exceed 100 bytes, which is equivalent to 50 two-byte Unicode characters. For a SQL Server database, the limit is 200 bytes, or 100 two-byte Unicode characters.

keys

The "keys" of an origin identifier contain the information necessary to reference the data in its original source. The pieces of information that you use to make up the keys differs depending on the source of the data. For data that originates in relational sources, you might use keys whose values include the source name, the table name, and the unique identifier of the data within that table.

The length of the origin identifier keys must not exceed the following sizes:

- On Db2: 1000 bytes. This is equivalent to 500 Unicode characters.
- On SQL Server: 692 bytes. This is equivalent to 346 Unicode characters.
- It is recommended that your origin identifiers are as short as possible in length, and that any common values are at the end of the key.
- Do not use non-printing or control characters in your origin identifiers because they might not be indexed correctly and cause your origin identifiers to be different from your intended values.

Creating origin identifiers for your data

There are two mechanisms for specifying the data for your origin identifiers. You can populate the staging table with all the information required to create the origin identifiers, or you can use a [combination of information in the staging table and the ingestion mapping](#).

When you can provide all the information in the staging tables, there is less processing of the data during ingestion, which can improve ingestion performance.

All information in the staging table

If you can populate the staging table with all the information for your origin identifiers, you can use the `origin_id_type` and `origin_id_keys` columns to store this information. Populate the `origin_id_type` column with the type of your origin identifier. Populate the `origin_id_keys` column with a unique value that is already a composite of key values including the unique identifier from the source. When you use these columns, you must specify them in the ingestion mapping file that you use.

To ingest links, you must specify the origin identifiers at the end of the link. You specify the "to" end of the link in the `to_origin_id_type` and `to_origin_id_keys` columns, and the "from" end in `from_origin_id_type` and `from_origin_id_keys`.

When you specify all the information in the staging table, the origin identifier section of your ingestion mapping is more simple. For example:

```
...
<originId>
  <type>$(origin_id_type)</type>
  <keys>
    <key>$(origin_id_keys)</key>
  </keys>
</originId>
...
```

To specify the origin identifiers at the link ends:

```
...
<fromOriginId>
  <type>$(from_origin_id_type)</type>
  <keys>
    <key>$(from_origin_id_keys)</key>
  </keys>
</fromOriginId>

<toOriginId>
  <type>$(to_origin_id_type)</type>
  <keys>
    <key>$(to_origin_id_keys)</key>
  </keys>
</toOriginId>
...
```

Combination of information in the staging table and the ingestion mapping

If you cannot populate the staging table with all the information for your origin identifiers, you can use the `source_id` column in the staging table to contain the unique identifier from the source and provide any other keys and the origin identifier type in the ingestion mapping.

To ingest links, you must specify the origin identifiers at the end of the link. You specify the unique identifier of the "to" end of the link in the `to_source_id` column, and the "from" end in `from_source_id`. In the ingestion mapping, you specify the other keys that make up the origin identifiers of the link ends.

When you provide the information in both the staging table and the ingestion mapping, the mapping file is more complex. For example:

```

...
<originId>
  <type>OI.EXAMPLE</type>
  <keys>
    <key>$(source_id)</key>
    <key>PERSON</key>
  </keys>
</originId>
...

```

To specify the origin identifiers at the link ends, if the to end is an "Account" entity type:

```

...
<fromOriginId>
  <type>OI.EXAMPLE</type>
  <keys>
    <key>$(from_source_id)</key>
    <key>PERSON</key>
  </keys>
</fromOriginId>

<toOriginId>
  <type>OI.EXAMPLE</type>
  <keys>
    <key>$(to_source_id)</key>
    <key>ACCOUNT</key>
  </keys>
</toOriginId>
...

```

The ingestInformationStoreRecords task

The parameters and usage of the `ingestInformationStoreRecords` depends on whether you are using the i2 Analyze deployment toolkit or the ETL toolkit.

- [i2 Analyze deployment toolkit](#)
- [ETL deployment toolkit](#)

i2 Analyze deployment toolkit

The deployment toolkit command for ingesting records looks like this:

```

setup -t ingestInformationStoreRecords
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p importMode=STANDARD|VALIDATE|BULK

```

Here, *ingestion_mapping_file* is the path to the XML file that contains the mapping that you want to use, and *ingestion_mapping_id* is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The `importLabel`, `importConfigFile`, and `importMode` parameters are optional:

- When you specify `importLabel`, *ingestion_label* is a name that identifies a particular use of the command in the Information Store's `IS_Public.Ingestion_Deletion_Reports` view.
- When you specify `importConfigFile`, *ingestion_settings_file* is the path to a settings file that contains *name=value* pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- When you specify `importMode`, you can set it to `STANDARD`, `VALIDATE`, or `BULK`.
 - `STANDARD` mode can be used to ingest new and updated records, with or without correlation. If you do not specify `importMode`, `STANDARD` mode is used.
 - `VALIDATE` mode checks the validity of the specified mapping, but no ingestion takes place.
 - `BULK` mode can be used to ingest new records without correlation.

ETL toolkit

The equivalent ETL toolkit command looks like this. The ETL toolkit command can also delete provenance from the Information Store.

```
ingestInformationStoreRecords
  -imf ingestion_mapping_file
  -imid ingestion_mapping_id
  -il ingestion_label
  -icf ingestion_settings_file
  -lcl true|false
  -im STANDARD|VALIDATE|BULK|DELETE_PREVIEW|DELETE|BULK_DELETE
```

Here, *ingestion_mapping_file* is the path to the XML file that contains the mapping that you want to use, and *ingestion_mapping_id* is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The other parameters are optional:

- When you specify `il`, *ingestion_label* is a name that identifies a particular use of the command in the Information Store's `IS_Public.Ingestion_Deletion_Reports` view.
- When you specify `icf`, *ingestion_settings_file* is the path to a settings file that contains *name=value* pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- When you specify `lcl` as `true`, any links that are removed as part of an entity delete operation are logged in `IS_Public.D<import identifier><entity type id>_<link type id>_Links` tables. For example, `IS_Public.D20180803090624143563ET5_LAC1_Links`.

You can specify `lcl` with the `DELETE` import mode only.

- When you specify `im` you can set it to one of the following modes:
 - `STANDARD` mode can be used to ingest new and updated records, with or without correlation. If you do not specify `im`, `STANDARD` mode is used.
 - `VALIDATE` mode checks the validity of the specified mapping, but no ingestion takes place.
 - `BULK` mode can be used to ingest new records without correlation.
 - `DELETE_PREVIEW` mode can be used to preview the effect of running a delete.

- DELETE mode can be used to delete provenance from the Information Store.
- BULK_DELETE mode can be used to delete provenance that does not contribute to a correlated record.

The previewDeleteProvenance and deleteProvenance tasks

To update the Information Store for deleted data in an external source, you update the Information Store by using staging tables and the deployment toolkit.

In the i2 Analyze deployment toolkit, you can use the **previewDeleteProvenance** and **deleteProvenance** toolkit tasks to update the Information Store. If you are using the ETL toolkit, you use the **ingestInformationStoreRecords** command with different import modes. For more information about using the ETL toolkit to remove provenance, see [“ETL toolkit” on page 407](#).

Because the delete process might cause significant numbers of i2 Analyze records to be deleted, two commands are provided. The first command previews the effect of running the second command before you commit to doing so. In the deployment toolkit, the two commands have different names but the same syntax:

```
setup -t previewDeleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
```

```
setup -t deleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p logConnectedLinks
      -p importMode=BULK_DELETE
```

Here, *ingestion_mapping_file* is the path to the XML file that contains the mapping that you want to use, and *ingestion_mapping_id* is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The other parameters are optional:

- When you specify `importLabel`, *ingestion_label* is a name that identifies a particular use of the command in the Information Store's `IS_Public.Ingestion_Deletion_Reports` view.
- When you specify `importConfigFile`, *ingestion_settings_file* is the path to a `settings` file that contains `name=value` pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- When you specify `logConnectedLinks`, any links that are removed as part of an entity delete operation are logged in `IS_Public.D<import identifier><entity type id>_<link type id>_Links` tables. For example, `IS_Public.D20180803090624143563ET5_LAC1_Links`.

You can specify `logConnectedLinks` when you do not specify `importMode` only.

- When you specify `importMode`, you can set it to `BULK_DELETE`. `BULK_DELETE` mode can be used to delete provenance that does not contribute to a correlated record. If you do not specify `importMode`, the standard delete process is used.

Understanding ingestion reports

Every attempt to add, update, or delete data in the Information Store through the deployment or ETL toolkit adds rows to the `IS_Public.Ingestion_Deletion_Reports` view. You can use the contents of this view to track the history of all such operations, and to examine the impact of a particular operation.

Each time that you run a command that might change the contents of the Information Store, you create a job in the database. Each job acts on one or more batches of i2 Analyze records. There is always one batch per item type that the command affects, but there can also be several batches for the same type if the number of affected records is large.

For example, consider a command that processes updates for deleted Person entity data. The first batch in the resulting job is for Person records, and there might be more such batches if there are many records to be deleted. If the Person data has links, then the job has further batches for each type of link that might get deleted as a result of the entity deletion.

The `IS_Public.Ingestion_Deletion_Reports` view contains information about every batch from every toolkit operation to create or update data in the Information Store. When you query the view, include **ORDER BY job_id** to group entries for the same job.

Note: Deletion-by-rule operations also result in job and batch creation, and view population, according to the same rules. For more information, see the *Deletion Guide*.

The first few columns in the view have the same value for all batches within a job:

Column name	Description
label	The value that you passed in the <code>importLabel</code> parameter of a toolkit command, or the value that a deletion-by-rule operation generates, or null.
job_id	The server-assigned identifier for this ingestion or deletion job. This identifier is also a cross-reference to the <code>Deletion_By_Rule_Log</code> view if the job originated from a deletion-by-rule operation.
ingestion_mode	The value that you passed in the <code>importMode</code> parameter, or Delete for all deletion-by-rule operations.
validation_mode	A description of how the job was configured to react to errors during the operation.
error_threshold	The threshold that applies to some of the validation modes.
primary_item_type	The i2 Analyze schema ID of the item type that was specified at job creation.
primary_record_count	The number of records of the primary item type that were affected by the job. (Deleting entity data can affect link records too.)
start_time	The start time of the job as a whole.
end_time	The end time of the job as a whole.

The remaining columns can have different values for different batches of records:

Column name	Description
batch_item_type	The i2 Analyze schema ID of the item type that was acted on in this batch. For at least one batch, the batch_item_type is the same as the primary_item_type.
batch_start_time	The start time of this batch, which is always later than the start time of the job.
batch_end_time	The end time of this batch, which is always earlier than the end time of the job.
insert_count	The number of rows of data from this batch that were inserted to the Information Store, resulting in new i2 Analyze records.
update_count	The number of rows of data from this batch that updated existing records in the Information Store.
merge_count	The number of <i>merge</i> operations that occurred in the Information Store from this batch.
unmerge_count	The number of <i>unmerge</i> operations that occurred in the Information Store from this batch.
delete_count	The number of pieces of provenance that were deleted from the Information Store as a result of this batch.
delete_record_count	The number of records that were deleted from the Information Store as a result of this batch.
reject_count	The number of rows that were rejected from this batch during processing because they are invalid.
status	An indicator of the result of this batch, from success (all rows processed correctly) through partial success to failure (no rows processed).
reject_view	The full name of the view that contains details of any rejected rows.
stack_trace	If i2 Analyze generated a stack trace as a result of errors during ingestion or deletion, this column contains it.

Ingest example

The (abbreviated) report for successful ingestion operations might look like this:

job_id	1	2
ingestion_mode	Standard	Standard
primary_item_type	ET10	ET4
primary_record_count	62	8
batch_item_type	ET10	ET4
batch_start_time	2017-11-30 15:27:06.76	2017-11-30 15:27:09.45
batch_end_time	2017-11-30 15:27:09.87	2017-11-30 15:27:09.63

insert_count	57	7
update_count	0	0
merge_count	5	1
unmerge_count	0	6
delete_count	0	0
delete_record_count	0	0
reject_count	0	0
status	Succeeded	Succeeded

In this example, several commands to ingest entity records resulted in the creation of several jobs. Each job demonstrates different behavior that is possible during ingestion, including correlation operations:

JOB_ID 1

This job demonstrates what the ingestion report can look like when data in the staging table causes merge operations. In this example, five merge operations are completed on the incoming rows of data, as shown in the `merge_count` column. This results in 57 i2 Analyze records created from the 62 rows of data, as shown in the `insert_count` and `primary_record_count` columns. This includes merging five rows of data with existing i2 Analyze records in the Information Store.

JOB_ID 2

This job demonstrates what the ingestion report can look like when the data in the staging table causes unmerge and merge operations. In this example, six unmerge operations are completed on the incoming rows of data, as shown in the `unmerge_count` column. One merge operation is completed on the incoming rows, as shown in the `merge_count` column. This results in 7 i2 Analyze records created, from eight rows of data as shown in the `insert_count` and `primary_record_count` columns. The `primary_record_count` value does not include the `unmerge_count`.

Delete example

The (abbreviated) report for a successful delete operation might look like this:

job_id	26	26	26	26	26
ingestion_mode	Delete	Delete	Delete	Delete	Delete
primary_item_type	ET5	ET5	ET5	ET5	ET5
primary_record_count	324	324	324	324	324
batch_item_type	ET5	LAC1	LAS1	LEM1	LIN1
batch_start_time	2017-11-30 15:27:06.76	2017-11-30 15:27:08.60	2017-11-30 15:27:08.60	2017-11-30 15:27:09.43	2017-11-30 15:27:09.45
batch_end_time	2017-11-30 15:27:09.87	2017-11-30 15:27:09.30	2017-11-30 15:27:09.29	2017-11-30 15:27:09.62	2017-11-30 15:27:09.63

insert_count	0	0	0	0	0
update_count	0	0	0	0	0
merge_count	0	0	0	0	0
unmerge_count	0	0	0	0	0
delete_count	324	187	27	54	33
delete_record_count	320	187	27	54	33
reject_count	0	0	0	0	0
status	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded

In this example, a command to update the Information Store for deleted entity data (with item type ET5) resulted in the creation of a job with five batches. The first few columns of the Ingestion_Deletion_Reports view contain the same values for all batches in the same job. Later columns reveal how deleting entity records results in the deletion of connected link records (with item types LAC1, LAS1, LEM1, LIN1).

In one case, the delete_record_count value is less than the delete_count value. This is because some of the provenance to be deleted was associated with an i2 Analyze record that had more than one piece of provenance. An i2 Analyze record is deleted only when the last associated provenance is deleted.

Troubleshooting the ingestion process

The commands that you run during the ingestion process send information about their progress to the command line and a log file. If any command encounters errors or does not run to completion, you can read the output to help you to diagnose the problem.

When an ingestion process runs to completion, the final output from the command is a report of what happened to the Information Store. The reports appear on the command line and in the ingestion log at `toolkit\configuration\logs\importer\IBM_i2_Importer.log`. The three possible end states are success, partial success, and failure.

Success

If the ingestion command processed all of the rows in the staging table without error, then the Information Store reflects the contents of the staging table. The command reports success like this example:

```
> INFO [IImportLogger] - Total number of rows processed: 54
> INFO [IImportLogger] - Number of records inserted: 0
> INFO [IImportLogger] - Number of records updated: 54
> INFO [IImportLogger] - Number of merges: 0
> INFO [IImportLogger] - Number of unmerges: 0
> INFO [IImportLogger] - Number of rows rejected: 0
> INFO [IImportLogger] - Duration: 5 s
> INFO [IImportLogger] -
> INFO [IImportLogger] - Result: SUCCESS
```

Partial success

If you ran the command in [record-based failure mode](#), and it processed some of the rows in the staging table without error, then it reports partial success like this example:

```
> INFO [IImportLogger] - Total number of rows processed: 34
> INFO [IImportLogger] - Number of records inserted: 0
> INFO [IImportLogger] - Number of records updated: 30
> INFO [IImportLogger] - Number of merges: 0
> INFO [IImportLogger] - Number of unmerges: 0
> INFO [IImportLogger] - Number of rows rejected: 4
> INFO [IImportLogger] - Duration: 4 s
> INFO [IImportLogger] -
> INFO [IImportLogger] - Result: PARTIAL SUCCESS
> INFO [IImportLogger] -
> INFO [IImportLogger] - Total number of errors: 4
> INFO [IImportLogger] - Error categories:
> INFO [IImportLogger] - ABSENT_VALUE: 4
> INFO [IImportLogger] -
> INFO [IImportLogger] - The rejected records and errors are recorded in
the database. For details, use the following view:
> INFO [IImportLogger] - IS_Staging.S20171204122426717092ET5_Rejects_V
```

The records in the Information Store reflect the rows from the staging table that the command successfully processed. The report includes the name of a database view that you can examine to discover what went wrong with each failed row.

Failure

If you ran the command in [mapping-based failure mode](#), then any error you see is the first one that it encountered, and the report is of failure:

```
> INFO [IImportLogger] - Total number of rows processed: 1
> INFO [IImportLogger] - Number of records inserted: 0
> INFO [IImportLogger] - Number of records updated: 0
> INFO [IImportLogger] - Number of merges: 0
> INFO [IImportLogger] - Number of unmerges: 0
> INFO [IImportLogger] - Number of rows rejected: 0
> INFO [IImportLogger] - Duration: 0 s
> INFO [IImportLogger] -
> INFO [IImportLogger] - Result: FAILURE
```

When the process fails in this fashion, the next lines of output describe the error in more detail. In this event, the command does not change the contents of the Information Store.

Note: If a serious error occurs, it is possible for the ingestion command not to run to completion. When that happens, it is harder to be certain of the state of the Information Store. The ingestion process uses batching, and the records in the store reflect the most recently completed batch. If you are using the bulk import mode, see [“Bulk import mode error”](#) on page 415 for more information about recovering from errors at this stage.

If the command reports partial success, you might be able to clean up the staging table by removing the rows that were ingested and fixing the rows that failed. However, the main benefit of record-based failure is that you can find out about multiple problems at the same time.

The most consistent approach to addressing failures of all types is to fix up the problems in the staging table and run the ingestion command again. The following sections describe how to react to some of the more common failures.

Link rows in the staging table refer to missing entity records

When the Information Store ingests link data, you might see the following error message in the console output:

```
Link data in the staging table refers to missing entity records
```

This message is displayed if the entity record at either end of a link is not present in the Information Store. To resolve the error:

- Examine the console output for your earlier operations to check that the Information Store ingested all the entity records properly.
- Ensure that the link end origin identifiers are constructed correctly, and exist for each row in the staging table.
- Ensure that the link type and the entity types at the end of the links are valid according to the i2 Analyze schema.

Then, rerun the ingestion command.

Rows in the staging table have duplicate origin identifiers

During any ingestion procedure, but especially when a staging table is large, you might see the following error message in the console output:

```
Rows in the staging table have duplicate origin identifiers
```

This message is displayed when several rows in a staging table generate the same origin identifier. For example, more than one row might have the same value in the `source_id` column.

If more than one row in the staging table contains the same provenance information, you must resolve the issue and repopulate the staging table. Alternatively, you can separate the rows so that they are not in the same staging table at the same time.

This problem is most likely to occur during an update to the Information Store that attempts to change the same record (with the same provenance) twice in the same batch. It might be appropriate to combine the changes, or to process only the last change. After you resolve the problem, repopulate the staging table and rerun the ingestion command.

Geospatial data is in the incorrect format

During an ingestion procedure that contains geospatial data, you might see the following error messages in the console output:

On Db2:

```
SQLERRMC=GSEGEOMFROMWKT;;GSE3052N Unknown type "F00(33.3" in WKT.
```

On SQL Server:

```
System.FormatException: 24114: The label F00(33.3 44.0) in the input well-known text (WKT) is not valid.
```

This message is displayed when data in a geospatial property column is not in the correct format.

Data in geospatial property columns must be in the `POINT(longitude latitude)` format. For more information, see [“Information Store property value ranges” on page 395](#).

Error occurred during a correlation operation

During an ingestion procedure with correlated data, you might see the following error message in the console output:

An error occurred during a correlation operation. There might be some data in an unusable state.

This message is displayed if the connection to the database or Solr is interrupted during a correlation operation.

To resolve the problem, you must repair the connection that caused the error, and then run the `syncInformationStoreCorrelation` toolkit task. This task synchronizes the data in the Information Store with the data in the Solr index so that the data returns to a usable state.

After you run the `syncInformationStoreCorrelation` task, reingest the data that you were ingesting when the failure occurred. Any attempt to run an ingestion or a deletion command before you run `syncInformationStoreCorrelation` will fail.

Ingestion with correlated data is still in progress

During an ingestion procedure, you might see the following error message in the console output:

You cannot ingest data because an ingestion with correlated data is still in progress, or because an error occurred during a correlation operation in a previous ingestion.

If another ingestion is still in progress, you must wait until it finishes. If a previous ingestion failed during a correlation operation, you must run the `syncInformationStoreCorrelation` toolkit task.

For more information about running the `syncInformationStoreCorrelation` toolkit task, see [“Error occurred during a correlation operation”](#) on page 415.

Ingestion of the same item type is still in progress

During an ingestion procedure, you might see the following error message in the console output:

You cannot ingest data for item type <ET5> because an ingestion is still in progress. You must wait until the process is finished before you can start another ingestion for this item type.

If another ingestion of the same item type is still in progress, you must wait until it finishes.

If you are sure that the ingestion is complete or not in progress, you can remove the file that is blocking the ingestion. To determine whether an ingestion is in progress, a file is created in the temporary directory on the server where the ingestion command was run. For example, `AppData\Local\Temp`. The file name is `INGESTION_IN_PROGRESS_<item type ID>`. After you remove the file, you can run the ingestion command again.

Bulk import mode error

The symptoms of this type of failure are a stack trace and failure message in the console and importer log. To recover from a failure at this time:

1. Identify the cause of the failure. You must use the SQL error codes to determine the cause of the failure.

You might see error messages about the following issues:

- Log size or connectivity issues.
- Invalid data in the staging table.

2. Fix the problem that caused the failure. This might include ensuring connectivity to the database or increasing the log size.

3. After you resolve the problem that caused the error, you can attempt the ingestion again. If any of the rows in the staging table were already ingested into the Information Store, you must remove them from the staging table before you can ingest in bulk mode.

- In the console or importer log, if the value for `Number of rows accepted` is 0 then run the ingestion command again.
- In the console or importer log, if the value for `Number of rows accepted` is greater than 0, you must ensure that these records are not ingested again.

Before you run the ingestion command again, add the `CheckExistingOriginIds=filter` setting to the import configuration file. When this value is set, the ingestion process calculates whether the origin identifiers in the staging table already exist in the Information Store and does not attempt to ingest them again.

When this is set to `filter`, the ingestion might take longer to complete. After the ingestion that failed is complete, you can remove the `CheckExistingOriginIds` setting from your import configuration file for future ingestion operations.

For more information about creating an import configuration file, see [“References and system properties” on page 401](#).

Information Store data correlation

This documentation provides an overview of the correlation functions that are available during data ingestion into the Information Store in IBM i2 Analyze. Later sections describe how to configure correlation with an example use case.

Intended audience

This documentation is intended for users who want to correlate data as it is being ingested into the Information Store.

Users must understand how to ingest data into the Information Store. For more information, see [Information Store data ingestion](#).

Users must understand the i2 Analyze data model, and i2 Analyze record structure. For more information, see [Data in i2 Analyze records](#).

Overview of correlation

Correlation is the process of associating multiple pieces of data with each other based on strong identifiers. For the process of ingesting data into the Information Store, i2 Analyze can use correlation identifiers that you provide to determine how to process and represent data in i2 Analyze records.

Correlation in i2 Analyze

During the ingestion process for the Information Store, correlation can be used to determine when data that is ingested is associated with existing records, and must be represented by a single i2 Analyze record. Conversely, if data no longer represents the same object, the data can be represented by multiple i2 Analyze records. In i2 Analyze, these operations are known as *merge* and *unmerge*.

During the correlation process, an identifier is used to determine how each row of data is associated. You present the identifier to the Information Store with the other staging data during ingestion.

You can limit the use of correlation to data from a specific source, of certain item types, or per row of data ingested into the Information Store. You do not have to provide a correlation identifier for all the data that you ingest into the Information Store.

Correlation uses

You might want to use correlation when you are ingesting data that originates from disparate sources, which have common properties, or have the potential to represent the same real-world objects. For example, if you have two data sources that contain information about people.

Another scenario where you might use correlation, is when the data that you are ingesting is in the form of event driven models (crime or complaint reports) where the same actors (people, locations, phones, and vehicles) might be referred to frequently in the same source.

Correlation can be used in these scenarios to combine multiple source records into single i2 Analyze records for link analysis.

Correlation method

In i2 Analyze, correlation identifiers and implicit discriminators are used to determine how the Information Store processes data during ingestion.

When you ingest data into the Information Store, you can provide a correlation identifier type and key value that are used to construct the correlation identifier for each row of data in the staging table. The type and key values that you provide are used to process data that is determined to represent the same real world object. Implicit discriminators are formed from parts of the i2 Analyze data model in the Information Store. Even if correlation identifiers match, if values for elements of the i2 Analyze data model are not compatible, that data cannot be represented by the same i2 Analyze record. For more information about correlation identifiers and implicit discriminators, see [“Correlation identifiers”](#) on page 417.

During the ingestion process, i2 Analyze compares the correlation identifiers of the data to be ingested and existing data in the Information Store. The value of the correlation identifiers determines the operations that occur. For more information about the correlation operations that can occur, see [“Correlation operations”](#) on page 419.

The example data sets demonstrate the correlation behavior in this release of i2 Analyze. For more information, see [“Ingesting example correlation data”](#) on page 435.

Correlation identifiers

The role of a correlation identifier is to indicate that data is about a specific real-world object. If multiple pieces of data are about the same specific real-world object, they have the same correlation identifier. At ingestion time, the correlation identifier of incoming data informs the Information Store how to process that data. Depending on the current state of the i2 Analyze record that is associated with the incoming data, a match with the correlation identifier on an inbound row of data determines the outcome of the association.

You specify the values for the correlation identifier in the staging table that you are ingesting the data from. The correlation identifier is made up of two parts, the *correlation identifier type* and the *correlation identifier key*.

type

The *type* of a correlation identifier specifies the type of correlation key that you are using as part of the correlation identifier. If you are generating correlation keys by using different methods, you might want to distinguish them by specifying the name of the method as the correlation identifier type. If your correlation keys are consistent regardless of how they are created, you might want to use a constant value for the correlation identifier type.

When you specify the identifier type, consider that this value might be seen by analysts.

The length of the value for the *type* must not exceed 100 bytes. This value is equivalent to 50 Unicode characters.

key

The *key* of a correlation identifier contains the information necessary to identify whether multiple pieces of data represent the same real-world object. If multiple pieces of data represent the same real-world object, they have the same correlation identifier key.

The length of the correlation identifier key must not exceed the following sizes:

- On Db2: 1000 bytes. This is equivalent to 500 Unicode characters.
- On SQL Server: 692 bytes. This is equivalent to 346 Unicode characters.

To prepare your data for correlation by i2 Analyze, you might choose to use a matching engine or context computing platform. Matching engines and context computing platforms can support the identification of matches that enable you to identify when data that is stored in multiple sources represents a single entity. You can provide these values to the Information Store at ingestion time. An example of such a tool is IBM InfoSphere Identity Insight. InfoSphere Identity Insight provides resolved entities with an entity identifier. If you are using such a platform, you can populate the correlation identifier type to record this, for example `identityInsight`. You might populate the correlation identifier key with the entity identifier, for example `1234`. This generates a correlation identifier of `identityInsight.1234`. For more information about IBM InfoSphere Identity Insight, see [Overview of IBM InfoSphere Identity Insight](#).

Alternatively, as part of the data processing to add data to the staging tables, you might populate the correlation identifier with values from property fields that distinguish entities. For example, to distinguish People entities you might combine the values for their date of birth and an identification number, and you might specify the type as `manual`. This generates a correlation identifier of `manual.1991-02-11123456`.

The complete correlation identifier is used for comparison. Only data with correlation identifiers of the same type is correlated.

For more information about specifying a correlation identifier during the ingestion process, see [Information Store staging tables](#).

Implicit discriminators

In addition to the correlation identifier that is created from the type and key values that you provide, *implicit discriminators* are also used during the matching process. In addition to the correlation identifier, the following implicit discriminators are also compared. The implicit discriminators must be compatible to enable correlation to occur.

Item type

The item type of the data that you are ingesting must be the same as the item type of the existing i2 Analyze record that is matched by the correlation identifier. If the item types are not the same, then no correlation operations occur.

Security dimension values

The security dimension values of the data that you are ingesting must be the same as the data that is matched by the correlation identifier. If the security dimension values are not the same, then no correlation operations occur.

Link direction and ends

For link data, the link direction and ends of the data that you are ingesting must be the same as the data that is matched by the correlation identifier. If the link direction and ends are not the same, then no correlation operations occur.

The direction and ends of a link are inspected, and direction is respected. For example, a link from A to B of direction 'WITH' matches with a link from B to A of direction 'AGAINST'. A link from A to B of direction 'WITH' does not match with a link from A to B of direction 'AGAINST'.

Correlation operations

When the Information Store receives a correlation identifier, the way the system responds depends on the value of the correlation identifier, and the state of any records that are associated with it.

The following section explains the *merge* and *unmerge* operations that the system can respond with when it receives a correlation identifier. This response is in addition to the insert and update operations that are part of the standard ingestion process.

Merge

When one or more pieces of data are determined to represent the same real-world object, the data is merged into a single i2 Analyze record. For the Information Store to merge data, the correlation identifiers must match, the implicit discriminators must be compatible, and the origin identifiers must be different.

During ingestion, a merge operation can occur in the following scenarios:

- New data in the staging table contains the same correlation identifier as an existing record in the Information Store. The new data has an origin identifier that is not associated with the existing record.
- An update to an existing record with a single piece of provenance in the Information Store causes the correlation identifier of that record to change. The new correlation identifier matches with another record in the Information Store.
- Multiple rows of data in the staging table contain the same correlation identifier. The Information Store ingests the data as a new i2 Analyze record, or the record merges with an existing record in the Information Store.

After a merge operation, the following statements are true for the merged record:

- The record has a piece of provenance for all of the source information that contributed to the merged record.
- By default, the property values for the merged i2 Analyze record are taken from the source information associated with the provenance that has the most recent value for the `source_last_updated` column.

If only one piece of provenance for a record has a value for the `source_last_updated` column, the property values from the source information that is associated with that provenance are used. Otherwise, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The piece of provenance that is last in the order is chosen. To ensure data consistency, update your existing records with a value for the `source_last_updated` column before you start to use correlation, and continue to update the value.

If the default behavior does not match the requirements of your deployment, you can change the method for defining property values for merged records. For more information, see [“Define how property values of merged records are calculated”](#) on page 425.

- If an existing record to be merged contained any notes, the notes are moved to the merged record.
- If an existing record to be merged was an entity record at the end of any links, the links are updated to reference the merged record.

Note: Any links that were created through Analyst's Notebook Premium are also updated to reference the merged record.

During ingestion, the number of merge operations that occur is reported in the merge_count column of the ingestion report.

The following diagrams demonstrate the merge operation.

In the first example of a merge operation, data in the staging table is merged into an existing i2 Analyze entity record because the correlation identifiers match and the origin identifiers are different.

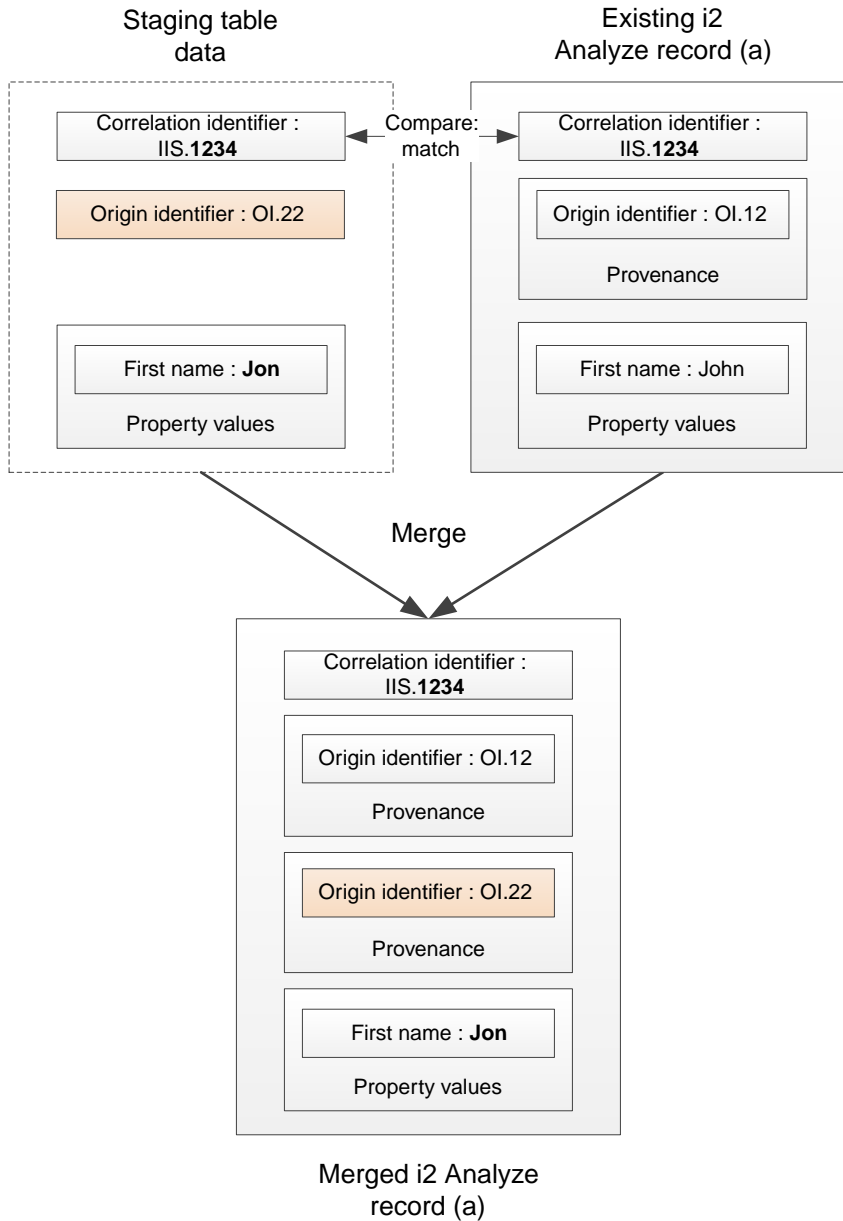


Figure 1. Incoming staging data merges with an existing i2 Analyze record.

In the diagram, the correlation identifiers of data in the staging table and the existing i2 Analyze record (a) match, which causes a merge operation. The existing i2 Analyze record (a) is not associated with

the origin identifier of the incoming data. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the data in the staging table row are used. This results in a change to the value for the first name property from "John" to "Jon". The merged i2 Analyze record (a) now contains provenance for the origin identifier 0I.12 and one for the new data, 0I.22.

In the second example of a merge operation, data in the staging table causes an update to an existing record (a) that changes the correlation identifier to match another record (b) in the Information Store, causing a merge.

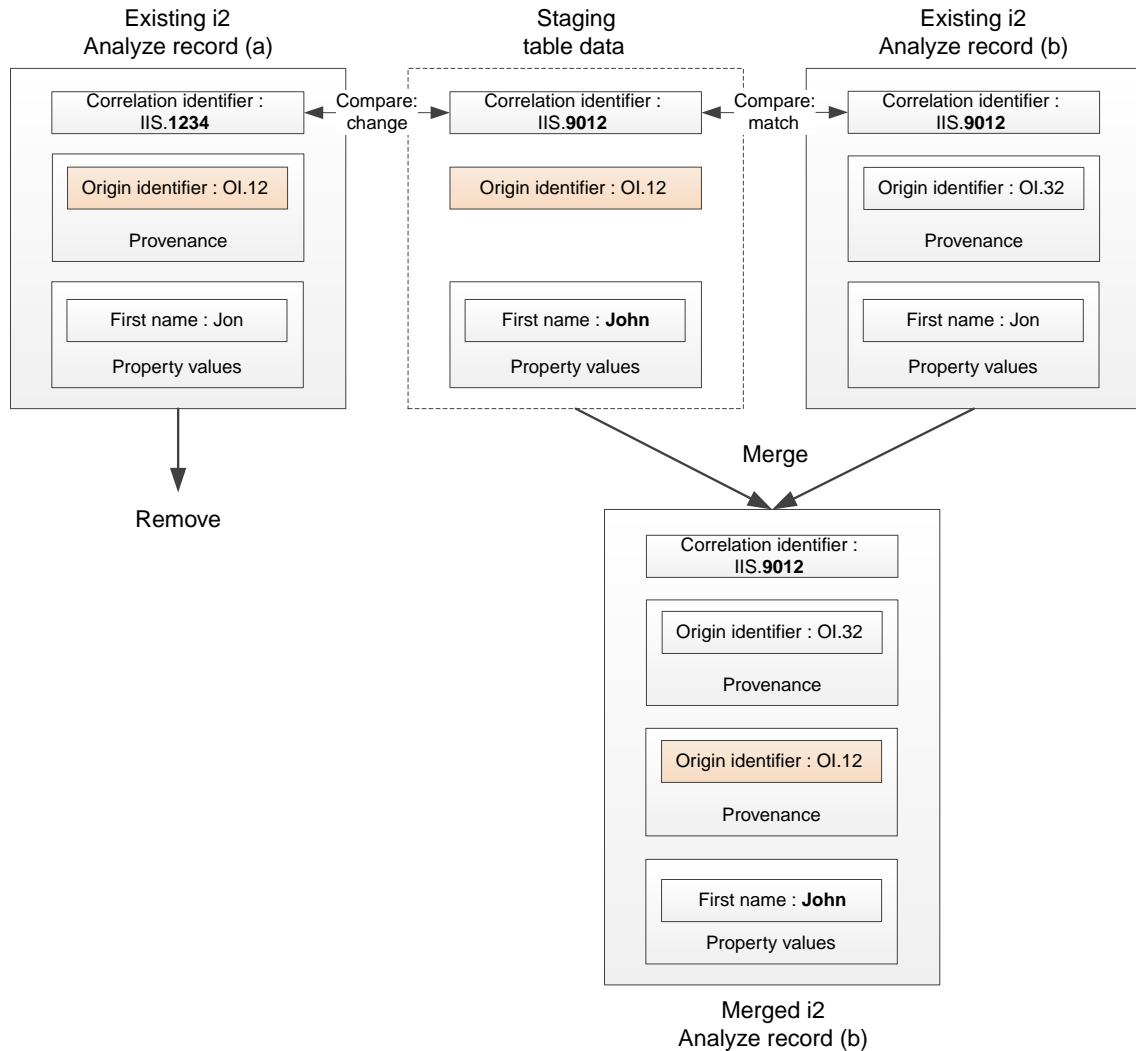


Figure 2. Incoming data updates an existing record, which causes the existing record to merge with another existing record in the Information Store. One of the existing records now has no provenance, and it is removed.

In the diagram, the data in the staging table has a different correlation identifier to the record (a) that it is currently associated with by its origin identifier, and the same correlation identifier as another existing record (b). This causes a merge operation. The first existing record (a) no longer has any

provenance associated with it, and is removed. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the staging table row are used. This results in a change to the value for the first name property from "Jon" to "John". The merged i2 Analyze record (b) contains multiple pieces of provenance, one for the origin identifier 0I.32 and one for the new data, 0I.12.

Unmerge

If the data for a merged i2 Analyze record is determined to no longer represent the same real-world object, the i2 Analyze record can be unmerged into two i2 Analyze records. For the Information Store to unmerge records, the correlation identifier or implicit discriminators of the data associated with that record must be changed.

Assuming that the implicit discriminators are compatible, the unmerge operation occurs when the correlation identifier of a row in the staging table is different from the correlation identifier on the merged record that it is currently associated with by its origin identifier.

After the unmerge operation, the following statements are true for the existing i2 Analyze record:

- The piece of provenance for the source information that caused the operation is unmerged from the record.
- The property values for the record are taken from the source information associated with the provenance that has the most recent value for `source_last_updated`.

If only one piece of provenance for a record has a value for the `source_last_updated` column, the property values from the source information that is associated with that provenance are used. Otherwise, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The piece of provenance that is last in the order is chosen. To ensure data consistency, update your existing records with a value for the `source_last_updated` column before you start to use correlation, and continue to update the value.

If the default behavior does not match the requirements of your deployment, you can change the method for defining property values for merged records. For more information, see [“Define how property values of merged records are calculated”](#) on page 425.

- All notes remain on the record.
- The *last updated time* of the record is updated to the time that the unmerge operation occurred.
- If the existing record was an entity record at the end of any links, any links to the unmerged piece of provenance are updated to reference the record that now contains the provenance.

After the unmerge operation, depending on the change in correlation identifier that is presented to the Information Store, either a new i2 Analyze record is inserted or a merge operation is completed. During ingestion, this process is reported in the `unmerge_count`, `insert_count`, and `merge_count` columns of the ingestion report.

The following diagrams demonstrate the unmerge operation. In each diagram, the data that is ingested from the staging table contains a different correlation identifier to the one on the record that it is associated with by its origin identifier. One unmerge operation results in a new record, and one results in a merge operation.

In the first example of an unmerge, the correlation identifier of the provenance that is unmerged from an existing record (a) does not match with another correlation identifier in the staging table or the Information Store. A new i2 Analyze record (b) is created with the property values from the staging table.

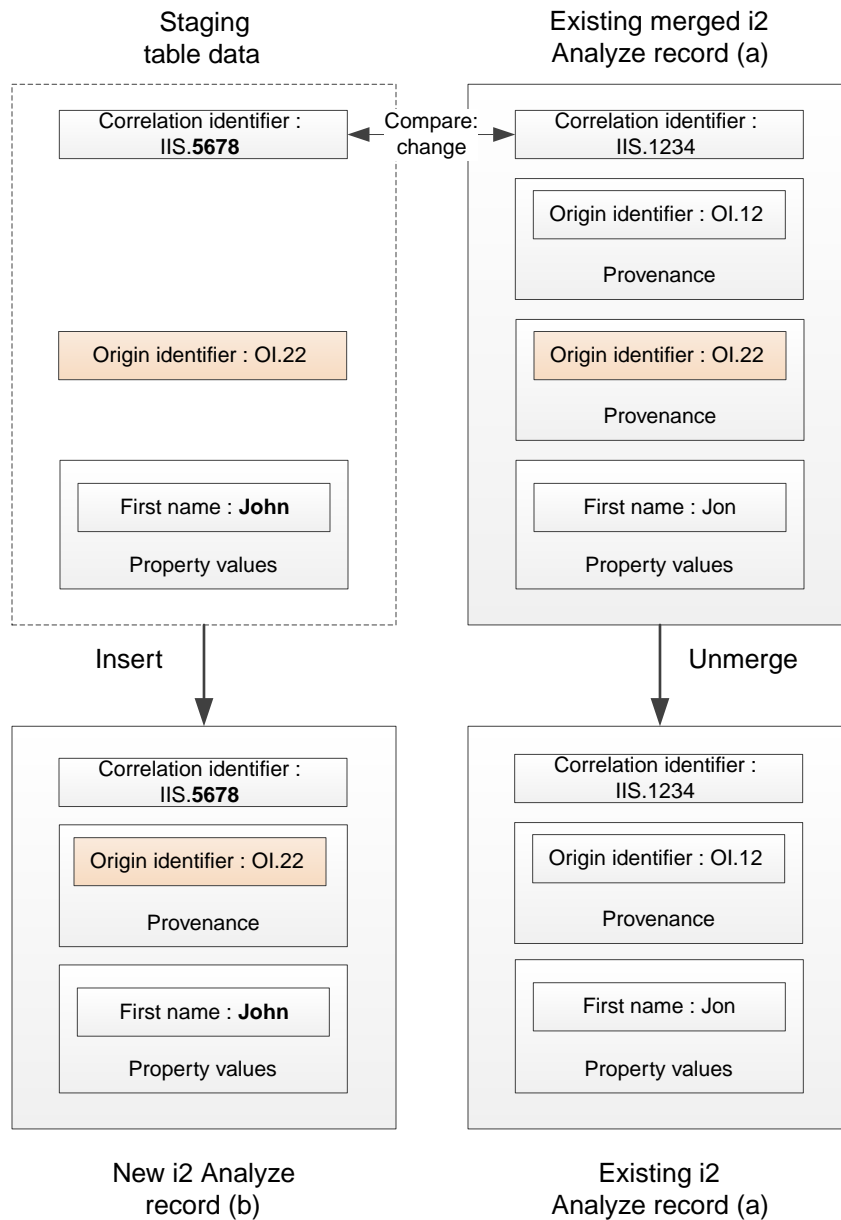


Figure 3. Incoming staging table data causes an unmerge operation. After the unmerge, a new record is inserted.

In the diagram, the data in the staging table has a different correlation identifier to the record (a) that it is currently associated with. This causes an unmerge operation. The provenance is unmerged from the existing record (a). The existing record (a) only contains the provenance for the origin identifier OI.12 and the property values from the source information that is associated with that provenance. The correlation identifier of the staging table data does not match with any others in the Information Store, so a new record (b) is inserted.

In the second example of an unmerge, if the correlation identifier of the provenance that is unmerged from an existing record (a) now matches the correlation identifier of another record (b), a merge operation is performed.

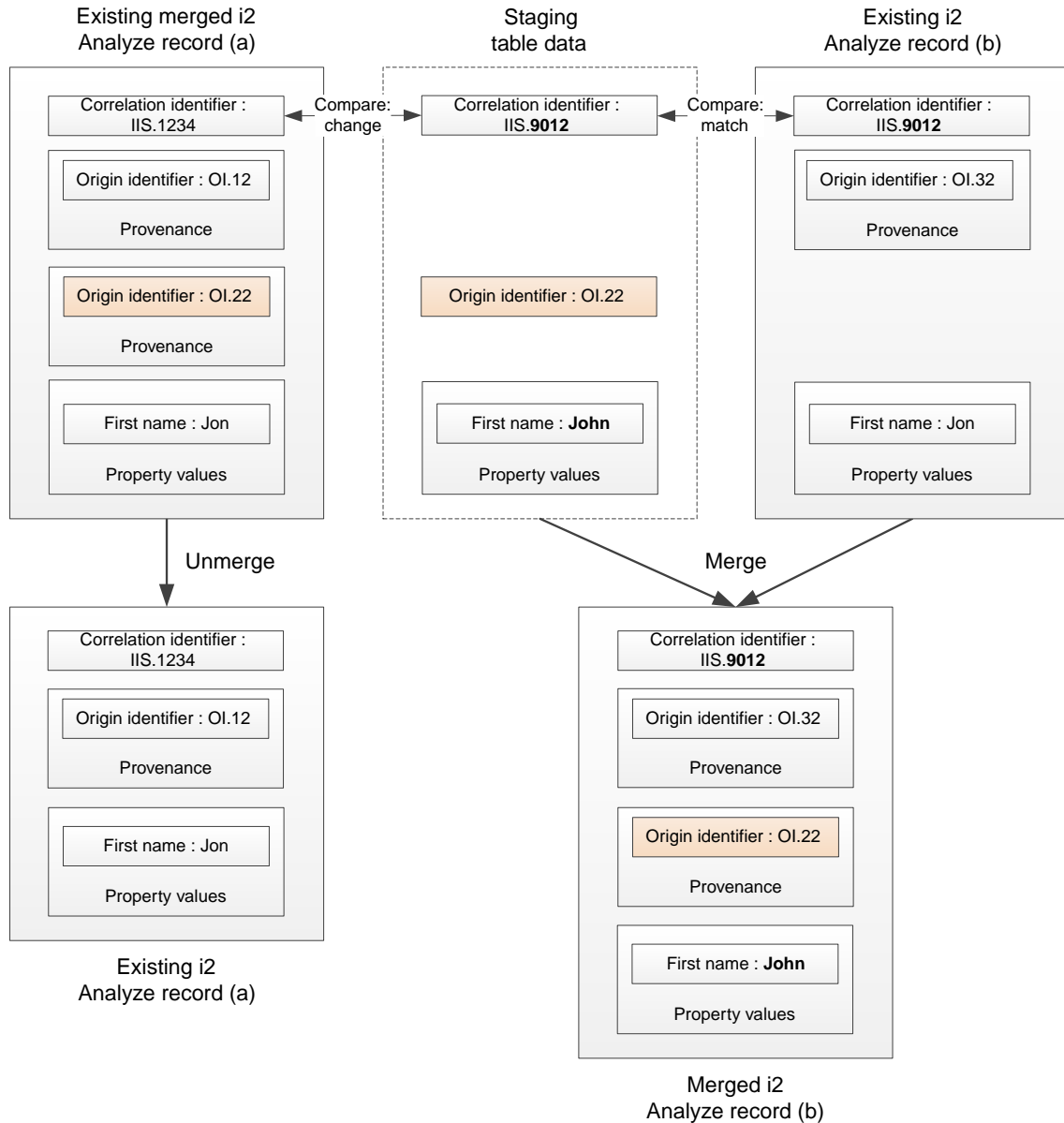


Figure 4. Incoming staging table data causes an unmerge operation. After the unmerge, a merge operation occurs.

In the diagram, the data in the staging table that is ingested has a different correlation identifier to the record (a) that it is currently associated with. This causes the unmerge operation. The origin identifier and provenance are unmerged from the existing record (a). The existing record (a) now only contains the provenance for the origin identifier OI.12 and the property values from the source information that is associated with that provenance.

The correlation identifier of the staging table data now matches with another record (b) in the Information Store, so a merge operation occurs. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the staging table row are used. This results in a change to the value for the first name property from "Jon" to "John". The merged i2 Analyze record (b) now contains two pieces of provenance, one for the origin identifier 0I.32 and one for the new data, 0I.22.

For more information about the behavior of a merge operation, see [“Merge” on page 419](#).

Hidden circular links

As a result of a merge operation, circular links might occur in the Information Store. The Information Store does not support circular links.

If data in the staging table causes existing records that are linked to each other in the Information Store to merge, the link becomes circular. As part of this operation, the existing link record is hidden in the Information Store. When a link record is hidden in the Information Store, analysts are unable to discover the link in search results and analysis.

A hidden link is revealed when the merged record at its ends is unmerged. After a hidden link is revealed, analysts are able to discover the link in search results and analysis.

Define how property values of merged records are calculated

In the Information Store, each property of an i2 Analyze record can have only one value. When multiple pieces of source data contribute to an i2 Analyze record, the system must calculate a single value for each property type.

Intended audience

The information about defining the property values of merged records is intended for users who are database administrators and experienced in SQL. To define the property values, you must write complex SQL view definition statements.

Important: You must write and test the SQL view statements for defining the property values of merged records in a non-production deployment of i2 Analyze. If you create an incorrect view, you might have to clear all the data from the system. Before you implement your view definitions in a production system, you must complete extensive testing in your development and test environments.

Why define the property values

When a record contains more than one piece of provenance, it is a merged record. The properties for an i2 Analyze record are calculated when a merge or unmerge operation occurs, or when provenance is removed from a record.

By default, all of the property values for a record come from the source data that contributed to the record with the most recent source-last-updated-time. If no source data has a source-last-updated-time, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The source data that is last in the order is chosen.

If the default source-last-updated-time behavior does not match the requirements of your deployment, you can define how the property values are calculated for merged i2 Analyze records. You might define your own rules when multiple data sources contain values for different properties of an item type or one data source is more reliable for a particular item or property type.

To demonstrate when it is useful to define how to calculate the property values for merged records, imagine that the following two pieces of source data contributed to an i2 Analyze record of type Person:

Origin identifier	Correlation identifier	Ingestion source name	Source last updated	First given name
DVLA1234	II1	DVLA	12:20:22 09/10/2018	John
PNC5678	II1	PNC	14:10:43 09/10/2018	Jon

In the default behavior, the property values are used from the source data with the most recent value for the Source last updated column. The property values from the row with the ingestion source name of *PNC* are used for the merged i2 Analyze record, and the record gets the value of *Jon* for the first given name property.

If you know that data from the *DVLA* ingestion source is more reliable for this item type, you can define that source data with the value of *DVLA* for the ingestion source name takes precedence. By using this definition, the i2 Analyze record gets the value of *John* for the first given name property.

After you define this rule for the Person entity type, all future updates to the records of this item type take the property values from the *DVLA* ingestion source if it is present in any of the source data that contributed to a merged i2 Analyze record.

For more information about how to enable this function, and create your own rules, see [“Defining the property values of merged i2 Analyze records”](#) on page 426.

Defining the property values of merged i2 Analyze records

By default, the mechanism for controlling the property values of merged records is not enabled in a deployment of i2 Analyze. To define which property values are used for merged records, you must inform i2 Analyze that you intend to do this and then customize views in the Information Store database.

Important: You must write and test the SQL view statements for defining the property values of merged records in a non-production deployment of i2 Analyze. If you create an incorrect view, you might have to clear all the data from the system. Before you implement your view definitions in a production system, you must complete extensive testing in your development and test environments.

To inform i2 Analyze that you intend to define the property values of merged records, you must run the `enableMergedPropertyValues` toolkit task. You can take control of the property values for records of specific item types, or all item types in the i2 Analyze schema.

When you run the toolkit task, two views are created for each item type that you specify. The views are created in the `IS_Public` schema, and are named with the display name of each item type. The two views provide the mechanism for viewing the data that contributes to the merged i2 Analyze records, and for creating rules to calculate a single value for each property:

- The view whose name is suffixed with `_MCV` contains all the data that contributed to each merged i2 Analyze record of that item type. This is the merge contributors view. This view is designed for you to inspect the column names and property values from the data in the Information Store. You must not modify the merge contributors view.

For more information about the merge contributors view, see [“The merge contributors view”](#) on page 428.

- The view whose name is suffixed with `_MPVDV` is where you can define the rules for calculating the property values of merged records. This is the merged property values definition view. You can modify this view to define how the merged property values are calculated from the data in the merge contributors view.

When you generate the merged property values definition view, the default source-last-updated-time behavior is implemented in the view.

For more information about modifying the definition, and some examples, see [“The merged property values definition view”](#) on page 429.

For example, the Person entity type from the example law enforcement schema produces the `IS_Public.E_Person_MCV` and `IS_Public.E_Person_MPVDV` views.

1. Identify the item types that you want to define the property values for.

For more information about why you might want to do this, see [“Define how property values of merged records are calculated”](#) on page 425.

2. Run the `enableMergedPropertyValues` toolkit task for each of the item types that you identified in step 1.

a) Open a command prompt and navigate to the `toolkit/scripts` directory.

b) Run the `enableMergedPropertyValues` toolkit task:

For example, to create the views for the Person entity type from the example law enforcement schema, run the following command:

```
setup -t enableMergedPropertyValues -p schemaTypeId=ET5
```

You can run the toolkit task with `schemaTypeId=ItemTypeId` or without any arguments to create the views for all item types in the `i2 Analyze` schema. You can run the task multiple times with different item type identifiers.

3. Inspect the merge contributors view (`_MCV`) that is created in the `IS_Public` schema and identify any rules to create for calculating the property values of merged records.

For more information about the merge contributors view, see [“The merge contributors view”](#) on page 428.

4. Modify the SQL create statement for the merged property values definition view (`_MPVDV`) that is created in the `IS_Public` schema to create a view that implements the rules you identified in step 3.

For more information about modifying the definition, and some examples, see [“The merged property values definition view”](#) on page 429.

After you ingest data that causes correlation operations, ensure that the property values for the merged `i2 Analyze` records match your expectations by searching for them. If you can ingest data successfully, and the property values are correct in your testing, the merged property values definition view is correct. If the property values are not correct in all cases, you must continue to modify the merged property values definition view and test the results until your requirements are met.

It is recommended that you keep a backup of your complete merged property values definition view.

During the testing of your deployment, you might need to clear any test data from the system to ensure that your view is behaving correctly. For more information, see [Clearing data from the system](#).

You must maintain the merged property values definition view for the lifetime of your deployment. If you change the `i2 Analyze` schema file, you must update the SQL statement for your view to match any changes. For example, if you add a property type to an item type, the new property must be added to the SQL statement for your merged property values definition view. You do not need to update the merge contributors view. The merge contributors view is updated when the schema is updated so that you can inspect the column name for the new property. For more information, see [“How to maintain your merged property values definition view”](#) on page 435.

After you test the view definition in your non-production database, you can implement the view on your production database.

The merge contributors view

In the merge contributors view you can see the source data that contributes to merged i2 Analyze records. By inspecting the view, you can gain an understanding about the data in your system and identify any rules to create that change the way that the property values are calculated.

To understand the merge contributors view, you must first understand the identifiers that the Information Store and i2 Analyze records use. For more information about i2 Analyze identifiers, see [Identifiers in i2 Analyze records](#).

The merge contributors view contains the following details about each piece of source data:

item_id

Item identifiers are used to distinguish i2 Analyze records of a particular type uniquely within the Information Store. Multiple rows in the merge contributors view can have the same item identifier. Each row that has the same item identifier represents one piece of source data that contributed to the i2 Analyze record with that identifier.

In the merge contributors view, the item identifier is stored in the `item_id` column.

record_id

Record identifiers distinguish i2 Analyze records uniquely throughout the deployment. In the merge contributors view, there is a one-to-one mapping between record identifiers and item identifiers.

In the merge contributors view, the record identifier is stored in the `record_id` column.

origin_id_type and origin_id_keys

The origin identifier is used to reference data in its original source. Each piece of source data has a unique origin identifier.

In the merge contributors view, the origin identifier is stored in the `origin_id_type` and `origin_id_keys` columns.

source_last_updated

When data is ingested into the Information Store, you can provide values for when the data was last updated in the data source from which it originated. The values in this column are used for the default source-last-updated-time behavior.

In the merge contributors view, the last updated time is stored in the `source_last_updated` column.

Property values

There is a column for each property type in the i2 Analyze schema. Each column is the display name of the property type prefixed with `p_`.

For example, values for the First Given Name property type for a Person entity type from the example law enforcement schema are stored in the `p_first_given_name` column.

correlation_id_type and correlation_id_key

The correlation identifier is used to indicate that data is about a specific real-world object. For multiple pieces of data to contribute to the same i2 Analyze record, they must have the same correlation identifier.

In the merge contributors view, the correlation identifier is stored in the `correlation_id_type` and `correlation_id_key` columns.

ingestion_source_name

When data is ingested into the Information Store, you must provide a data source name that identifies the data source from which it originated. This is stored as the ingestion source name in the Information Store.

In the merge contributors view, the ingestion source name is stored in the `ingestion_source_name` column.

In most scenarios, the `origin_id_type`, `origin_id_keys`, `source_last_updated`, and `ingestion_source_name` values can be used for choosing the source data to provide the property values.

The following table shows an example merge contributors view for a Person entity type where two pieces of source data contribute to the same i2 Analyze record:

record_id	2R1HCAGm2FnmSkdgRbRjpTu W	2R1HCAGm2FnmSkdgRbRjpTu W
item_id	1	1
origin_id_type	DVLA	PNC
origin_id_keys	1234	5678
source_last_updated	12:20:22 09/10/2018	14:10:43 09/10/2018
p_first_give_name	John	Jon
p_middle_name	James	
p_...
correlation_id_type	II	II
correlation_id_key	1	1
ingestion_source_name	DVLA	PNC

In this example merge contributors view, you can see the following details:

- The data originates from two ingestion sources; DVLA and PNC
- The data from the PNC ingestion source was updated more recently than the data from the DVLA ingestion source
- The data from the PNC ingestion source is missing a value for the middle name property

You might already know some of these details about the data in your system, or you might be able to determine these differences from inspecting the contents of the view. If you decide that you want to change the default behavior for calculating property values, you must customize the merged property values definition view. For more information about how to customize the merged property values definition view, see [“The merged property values definition view”](#) on page 429.

The merged property values definition view

The merged property values definition view is used to calculate the property values of merged i2 Analyze records. You can modify the view to define which property values records receive.

The merged property values definition view is used to calculate a single value for each property type for each merged record.

After you inspect the merge contributors view and identify the rules that you want to define, you must modify the SQL statement to create a view that matches your requirements. For more information about the merge contributors view, see [“The merge contributors view”](#) on page 428.

After you create a merged property values definition view, you must maintain the view through the lifetime of your deployment. For more information, see [“How to maintain your merged property values definition view”](#) on page 435.

Requirements of the view

The merged property values definition view is over the merge contributors view. To produce a single value for each property type, the view must contain one row for each `item_id` and populate each column with a value. You can include NULL values for property types that have no value.

After you customize the merged property values definition view, it must conform to the following requirements:

- The view must have columns for the item identifier and every property type column in the merge contributors view. The column names in both views must be the same.

The merged property values definition view must not contain the `origin_id_type`, `origin_id_keys`, `source_last_updated`, `correlation_id_type`, `correlation_id_keys`, or `ingestion_source_name` columns.

- Each item identifier must be unique.

Important: You cannot ingest data into the Information Store if the view contains multiple rows with the same item identifier.

Default behavior

By default, the merged property values definition view uses the default source-last-updated-time behavior to define which source data the property values come from. You can see the SQL statement for the view in IBM Data Studio or SQL Server Management Studio. An example of the SQL statement for the generated view for the Person entity from the example law enforcement schema is:

```
SELECT item_id, p_unique_reference, p_title,
... (all property type columns) ...
p2_issued_date_and_time, p3_issued_date_and_time
FROM
(
  SELECT MCV.*,
         ROW_NUMBER ( ) OVER (PARTITION BY MCV.item_id
                              ORDER BY source_last_updated DESC NULLS LAST,
                                       origin_id_keys DESC,
                                       origin_id_type DESC)
         AS partition_row_number
  FROM IS_Public.E_Person_MCV AS MCV
  INNER JOIN IS_Public.E_Person_STP AS STP
           ON MCV.item_id = STP.item_id
) AS P
WHERE partition_row_number = 1
```

The SQL statement for the default view works in the following way:

- To ensure that the view contains the correct columns, the first SELECT statement returns the `item_id` and all property type columns. For example, `p_title`.
- To split the rows into groups that contain only the data that contributed to a single merged record, the `OVER` and `PARTITION BY` clauses are used on the `item_id` column.
- To define the value for each property, the `ROW_NUMBER` function and `ORDER BY` clauses are used. For the default behavior, the `ORDER BY` clause is used on the `source_last_updated`, `origin_id_keys`, and `origin_id_type` columns. Each column is in descending order. By using `NULLS LAST`, you ensure that if the `source_last_updated` column does not contain a value, it is listed last.

The `ROW_NUMBER` function returns the number of each row in the partition. After the rows are ordered, the first row is number 1. The initial SELECT statement uses a `WHERE` clause to select the values for each row with a value of 1 for `partition_row_number`. Because only one row has a value of 1, the view contains only one row for each item identifier.

- When you enable merged property views by using the `enableMergedPropertyViews` toolkit task, a table alias for an internal staging table is also created. For example, `IS_Public.E_Person_STP` is the table alias for the staging table that is created during a Person item type ingestion.

During the ingestion process, the internal staging table contains the item identifiers of the records to be inserted or updated as part of the current ingestion in the `item_id` column. When the property values for records are calculated by using the `_MPVDV` view, it uses the information in the `_STP` alias to restrict the number of rows that are processed. The following statement from the previous example ensures that the `_MPVDV` view uses the alias for a specific item type:

```
INNER JOIN IS_Public.E_Person_STP AS STP
      ON MCV.item_id = STP.item_id
```

Examples

You can modify the following example views to match the data in your i2 Analyze schema or to meet your property value requirements:

Ingestion Source Name precedence

In this example, the data is ordered so that a specified ingestion source takes precedence. For example, the *DVLA* ingestion source takes precedence over the *PNC* one.

```
SELECT item_id, p_unique_reference, p_title,
... (all property type columns) ...
p2_issued_date_and_time, p3_issued_date_and_time
FROM (
  SELECT MCV.*,
         ROW_NUMBER ( ) OVER (PARTITION BY MCV.item_id
                              ORDER BY CASE ingestion_source_name
                                       WHEN 'DVLA' THEN 1
                                       WHEN 'PNC' THEN 2
                                       ELSE 3
                              END
         ) AS partition_row_number
  FROM IS_Public.E_Person_MCV AS MCV
  INNER JOIN IS_Public.E_Person_STP AS STP
    ON MCV.item_id = STP.item_id
) AS P
WHERE partition_row_number = 1
```

If an ingestion source name does not match *DVLA* or *PNC*, it is ordered last. If multiple contributing pieces of data have the same ingestion source name, the behavior is non-deterministic. In a production system, you must include another clause that provides deterministic ordering in all scenarios.

To order the ingestion sources, the `ORDER BY CASE` clause is used. The rows are ordered by the value that they are assigned in the `CASE` expression. For more information about the `ORDER BY CASE` statement, for Db2 see [ORDER BY clause](#) and [CASE expression](#) or for SQL Server see [Using CASE in an ORDER BY clause](#).

The `ROW_NUMBER` function and the `OVER` and `PARTITION BY` clauses are used in the same way as the view for the default behavior.

Source last updated and non-NULL

In this example, the data is ordered by the source last updated time. However if there is a `NULL` value for a particular property type, the value is taken from the data that has the next most recent time. This process is completed until a value for the property is found, or no data is left for that record. The definitions are slightly different depending on the database management system that hosts the Information Store database.

Db2

```
SELECT DISTINCT
    item_id,

    FIRST_VALUE(p_first_given_name,
        'IGNORE NULLS')
        OVER ( PARTITION BY    MCV.item_id
              ORDER BY        source_last_updated DESC NULLS LAST
              RANGE BETWEEN   UNBOUNDED PRECEDING AND
                              UNBOUNDED FOLLOWING
              ) AS p_first_given_name,

... (FIRST_VALUE function for all property types) ...

    FIRST_VALUE(CAST(p_additional_informatio AS VARCHAR(1000)),
        'IGNORE NULLS')
        OVER ( PARTITION BY    MCV.item_id
              ORDER BY        source_last_updated DESC NULLS LAST
              RANGE BETWEEN   UNBOUNDED PRECEDING AND
                              UNBOUNDED FOLLOWING
              ) AS p_additional_informatio

FROM      IS_Public.E_Person_MCV
INNER JOIN IS_Public.E_Person_STP AS STP
          ON MCV.item_id = STP.item_id
```

SQL Server

```
SELECT DISTINCT
    P.item_id,
    P1.p_first_given_name,
    P2.p_additional_informatio,
    ... (all property type columns) ...
```

```

FROM "IS_Public".E_Person_MCV AS P
LEFT OUTER JOIN
( SELECT DISTINCT
  item_id,
  FIRST_VALUE(p_first_given_name)
    OVER ( PARTITION BY MCV.item_id
          ORDER BY CASE source_last_updated
                    WHEN NULL THEN
                      CAST('0001-01-01' AS date)
                    ELSE
                      source_last_updated
                    END DESC
          RANGE BETWEEN UNBOUNDED PRECEDING AND
                       UNBOUNDED FOLLOWING
        ) AS p_first_given_name
FROM   "IS_Public".E_Person_MCV
INNER JOIN IS_Public.E_Person_STP AS STP
  ON MCV.item_id = STP.item_id
WHERE   p_first_given_name IS NOT NULL
  ) AS P1
ON P1.item_id = P.item_id

```

... (LEFT OUTER JOIN and FIRST_VALUE function for all property types) ...

```

LEFT OUTER JOIN
( SELECT DISTINCT
  item_id,
  FIRST_VALUE(CAST(p_additional_informatio AS VARCHAR(1000)))
    OVER ( PARTITION BY MCV.item_id
          ORDER BY CASE source_last_updated
                    WHEN NULL THEN
                      CAST('0001-01-01' AS date)
                    ELSE
                      source_last_updated
                    END DESC
          RANGE BETWEEN UNBOUNDED PRECEDING AND
                       UNBOUNDED FOLLOWING
        ) AS p_additional_informatio
FROM   "IS_Public".E_Person_MCV
INNER JOIN IS_Public.E_Person_STP AS STP
  ON MCV.item_id = STP.item_id
WHERE   p_additional_informatio IS NOT NULL
  ) AS P2
ON P2.item_id = P.item_id

```

This example uses the FIRST_VALUE function to identify which data has the first non-NULL value for a property type. The FIRST_VALUE function returns the first value in an ordered set of values. You must specify a scalar expression, and PARTITION BY, ORDER BY, and RANGE clauses. In the example, the scalar expressions are the property type columns. The PARTITION BY and ORDER BY

clauses are similar to the other examples, where they act on the item identifier and the source last updated time.

In Db2, the FIRST_VALUE function allows you to ignore null values by specifying 'IGNORE NULLS' in the expression. This functionality is not available in SQL Server. To achieve the same result, you must perform a series of LEFT OUTER JOINS for each FIRST_VALUE function. In the ORDER BY CASE clause for each FIRST_VALUE function, when the value for the column in the FIRST_VALUE function is NULL you can set the source last updated time to 0001-01-01. Setting the last updated time to this value and ordering the partition this way ensures that the row with a value for the property and the most recent source last updated time is returned.

- For more information about the FIRST_VALUE function in Db2, see [OLAP specification - FIRST_VALUE](#).
- For more information about the FIRST_VALUE function in SQL Server see [FIRST_VALUE \(Transact-SQL\)](#).

To use the FIRST_VALUE function, you must define a function for each property type separately. In the previous examples, there are two examples of the FIRST_VALUE function for two property types:

- The first acts on the p_first_given_name column.
- The second acts on the p_additional_informatio column. The same process is completed, however in the merge contributors view the p_additional_informatio column is of data type CLOB(32M). This data type is not supported by the FIRST_VALUE function. To use this column, you must cast it into a supported data type. For example, VARCHAR(1000). You must ensure that if a column is cast, that you do not lose any data.

How to maintain your merged property values definition view

The merged property values definition views include all of the property types for an item type as they were when the view was created. If you change the i2 Analyze schema to add a property type to an item type, the merged property values definition view is now incorrect. The merged property values definition view does not contain a column for the new property type, and therefore cannot populate an i2 Analyze record with a value for that property. When you add property types to the i2 Analyze schema for an item type, you must update any merged property values definition view to include any new property types. i2 Analyze updates the merge contributors view when the schema is updated, you can use the updated merge contributors view to see the name of the new column.

Note: If you do not update your merged property values definition view, you cannot ingest data into the Information Store.

To stop controlling the property values of merged records for an item type, you can inform i2 Analyze that you no longer intend to use this function. Inform i2 Analyze by running the disableMergedPropertyValues toolkit task. You can specify `schemaTypeId=ItemTypeId` to remove the views for one item type only, or without any arguments to remove the views for every item type in the i2 Analyze schema. After you run the toolkit task, the default source-last-updated-time behavior is used.

You can modify the merged property values definition view or run the disableMergedPropertyValues toolkit task at any time. The property values of i2 Analyze records that are already in the Information Store are not updated. If you ingest any data that updates an existing i2 Analyze record, the property values for that record are recalculated with the new merged property values definition view.

Ingesting example correlation data

The correlation example provides sets of data that were passed through a matching engine so that each row of data in the set is associated with a correlation identifier. You can ingest the example data,

and then inspect the items in the Information Store from Analyst's Notebook Premium to demonstrate the correlation behavior.

Use the `ingestExampleData` toolkit task to ingest example data sets that demonstrate the correlation behavior in i2 Analyze. For more information about the operations that occur, and how the i2 Analyze records are effected, see [“Correlation operations”](#) on page 419.

law-enforcement-data-set-2

This data set contains data with correlation identifiers. After you ingest this data set, the Information Store database contains i2 Analyze records with correlation identifiers. No correlation operations occur when you ingest this data set.

law-enforcement-data-set-2-merge

This data set contains data that causes a number of merge operations to occur with some of the i2 Analyze records that were created from the first data set.

law-enforcement-data-set-2-unmerge

This data set contains data that causes a number of unmerge operations to occur with some of the i2 Analyze records that were merged from the second data set.

You can find the example data sets in the `toolkit\examples\data` directory.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Ingest the first example data set.
 - a) Run the following command to ingest the first example data set:

```
setup -t ingestExampleData -e law-enforcement-data-set-2
```

The `examples\data\law-enforcement-data-set-2` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables. Each row of data contains a correlation identifier type and a unique key value.

The Information Store now contains i2 Analyze records with correlation identifiers. However, no correlation operations occurred during the ingestion.

- b) In Analyst's Notebook Premium, search for "Julia Yochum" and add the returned entity to the chart.
3. Ingest the `law-enforcement-data-set-2-merge` data to demonstrate merge operations.
 - a) Run the following command to ingest the second example data set:

```
setup -t ingestExampleData -e law-enforcement-data-set-2-merge
```

The `examples\data\law-enforcement-data-set-2-merge` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables. Each row of data contains a correlation identifier type and key value. Each row contains a unique value for the `source_id`, which represents the origin identifier.

In this scenario, the matching engine identified that some of the data represents the same real-world objects as data in the first data set. As part of this process, the correlation identifiers match with some of the existing data in the database, but the origin identifiers are different. These matches cause a number of merge operations to occur during the ingestion.

For example, you can see on line 2 of the `person.csv` file the correlation identifier key is `person_0`, which matches the correlation identifier of the record for person "Julia Yochum". This match causes a merge between the existing record and the incoming row of data. As part of the merge, the property values from the incoming row of data are used for the record, this results in

the full name changing to "Julie Yocham". This is an example of the scenario that is described in [Figure 1 on page 420](#)

In the ingestion reports, you can see the number and type of correlation operations that occurred during the ingestion. For more information about understanding the ingestion reports, see [Understanding ingestion reports](#).

- b) In Analyst's Notebook Premium, select the chart item that represents Julia Yochum and click **Get changes**.

You can see that the name changes due to the merge operation described previously.

4. Ingest the law-enforcement-data-set-2-unmerge data to demonstrate unmerge operations.

- a) Run the following command to ingest the third example data set:

```
setup -t ingestExampleData -e law-enforcement-data-set-2-unmerge
```

The examples\data\law-enforcement-data-set-2-unmerge directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables.

In this scenario, the matching engine identified that some of the data no longer represents the same real-world objects as it did previously. As part of this process, the correlation identifiers of previously merged data are changed. These changes cause a number of unmerge operations to occur during the ingestion.

In this example, the correlation identifier of the data that was ingested has changed, but the origin identifier remained the same. For example, on line 2 of the person.csv file, the correlation identifier key is now person_1101 for origin identifier PER:GEN\1101. Before, this value was person_0. This causes an unmerge operation on the record that the data is currently associated with. This is an example of the scenario that is described in [Figure 3 on page 423](#)

You can see the number of unmerge operations that occurred in the ingestion reports.

- b) In Analyst's Notebook Premium, search for "Julie Yocham".

There are now two entities for Julie Yocham.

After you investigate the correlation behavior, you can clear the example data from your system. For more information, see [Clearing data from the system](#).

Information Store data deletion

About this guide

IBM i2 Analyze supports several different mechanisms for creating records in the Information Store. It also supports different mechanisms for deleting them. This document describes how to delete records individually, selectively, or in bulk; and it describes the circumstances in which each approach is possible or desirable.

This guide is intended for system implementers and database administrators who want to delete data directly from the Information Store. It applies to data added by ingestion from one or more external data sources, and to data added by users who share information through IBM i2 Analyst's Notebook Premium.

Note: For more information about adding data by ingestion, see [Information Store data ingestion](#). For more information about adding data through IBM i2 Analyst's Notebook Premium, see [Uploading records to the Information Store](#).

The tasks that the following topics describe variously require familiarity with the IBM i2 Analyze data model, your database management systems, SQL queries, and IBM Data Studio or SQL Server Management Studio. Some of the approaches also require specialized authorization with the Information Store.

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users to resolve problems with the product.

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM software subscription and support contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

To contact IBM Support about a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM Support topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
 - Online through the IBM Support Portal at <https://www.ibm.com/mysupport/>. You can open, update, and view all of your service requests from the Service Request portlet on the Service Request page.
 - By phone. For the phone number to call in your region, see the Directory of worldwide contacts web page at <https://www.ibm.com/planetwide/>

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

Overview

To comply with data protection regulations, storage restrictions, or other requirements, you might need to delete data from the Information Store. IBM i2 Analyze provides mechanisms for doing so. The mechanism that you choose depends on how the records were created in the Information Store and your reason for deleting them.

At version 4.3.5 of i2 Analyze, there are two ways to delete records from the Information Store:

- Some of the records in the Information Store are created and uploaded through Analyst's Notebook Premium. If a user deletes a record that they or a colleague uploaded, it becomes unreachable for all users, but remains in the Information Store. You can arrange to delete these *soft-deleted* records permanently, either automatically or on demand.
- For all records in the Information Store, you can write identifying rules that target them for deletion individually or as a group. This approach deletes records no matter how they were originally created or what their current state is.

It is common to all record deletion that deleting an entity record forces i2 Analyze to delete the link records that are attached to that entity. A link must always have two ends. If one end is deleted, the link is automatically deleted as well.

It is also common to both of these ways for deleting records from the Information Store that the procedure is permanent. If you might need to recover the deleted data in future, ensure that you have a backup plan in place before you begin.

Note: The demands of synchronizing with the contents of an external source can also require you to delete data from the Information Store. In that situation, you can reflect deleted data in an external source by using the same ingestion pipeline that you use to reflect creation and modification.

For more information about updating the Information Store in this manner, see [Updating the Information Store for deleted data](#).

Deleting shared records

Analyst's Notebook Premium enables authorized users to create and delete records in the Information Store. However, even authorized users can delete only records that they or their colleagues create, and the operation is not immediately permanent. To delete records completely, you must interact with the Information Store database.

When a user deletes a record in the Information Store through Analyst's Notebook Premium, the effect for all users is instant. The record is no longer returned in the results of search or expand operations, and copies of the record on other Analyst's Notebook charts become orphaned.

However, by default, the data for the record remains in the database until you decide to delete it permanently. The Information Store has a mechanism for removing these "soft-deleted" records that you can start on an automatic or a manual basis.

The mechanism for purging (that is, fully deleting soft-deleted) records uses three stored procedures from the IS_Public schema of the Information Store database.

Procedure	Parameters	Description
Purge_Soft_Deleted_Records		Permanently deletes all soft-deleted records of all types from the Information Store.
Set_Purge_Soft_Delete_Schedule	Schedule	Sets and activates the schedule on which soft-deleted records are automatically deleted from the Information Store. If you are using Db2, the parameter must be a string in UNIX cron format. For more information, see the UNIX cron format . If you are using SQL Server, you must use the arguments and values accepted by the sp_add_jobschedule .
Remove_Purge_Soft_Delete_Schedule		Clears and deactivates the automatic deletion schedule.

To do its work, the purging mechanism uses the same infrastructure as the mechanism for deleting records by rule. Each manual or automated request to purge soft-deleted records from the Information Store causes a set of deletion jobs to be created. There is one job for each entity type and link type in the i2 Analyze schema, and the Db2 task scheduler or SQL Server Agent runs them at the earliest opportunity.

i2 Analyze keeps logs of manual and automated purge operations alongside its logs of deletion-by-rule operations, in the “[Deletion_By_Rule_Log view](#)” on page 450. In that view, the rule name for all jobs that are associated with purging soft-deleted records is PURGE_SOFT_DELETE.

Purging soft-deleted records

Users who delete records from the Information Store through Analyst's Notebook Premium have no way to recover them afterward, but by default they are not completely removed. As the database administrator, you can decide to leave these soft-deleted records in the database, or to purge them manually or automatically.

Purging soft-deleted records from the Information Store requires the same authorization as deleting records by rule. You must connect to the database as a user with the “[Authorization to delete by rule](#)” on page 452. The following instructions assume that you are connected to the database as a user with that role.

In addition, if you want to automate purging, you must be able to schedule jobs to run on the database management system:

- If you are using IBM Db2, the Db2 administrative task scheduler must be enabled.
- If you are using Microsoft SQL Server, the SQL Agent service must be started.

For more information, see the IBM i2 Analyze [software prerequisites](#).

When users delete records from the Information Store through Analyst's Notebook, the data remains in the Information Store (but is inaccessible to users) unless you do something to change that. The following procedures describe how to purge soft-deleted records manually, how to automate that process, and how to understand the effect of a purge operation.

To perform a one-off, manual purge of all soft-deleted records from the Information Store:

- Run the `IS_Public.Purge_Soft_Deleted_Records` stored procedure.
 - For example, for Db2:

```
CALL IS_Public.Purge_Soft_Deleted_Records;
```

- For example, for SQL Server:

```
EXECUTE "IS_Public".Purge_Soft_Deleted_Records;
```

You can use this procedure regardless of whether you also configure automatic purging.

It immediately creates jobs for purging soft-deleted records of every type.

To set up the Information Store so that soft-deleted records are purged automatically, on a schedule:

- Run the `IS_Public.Set_Purge_Soft_Delete_Schedule` stored procedure with the schedule that you want to use.
 - For example, for Db2:

```
CALL IS_Public.Set_Purge_Soft_Delete_Schedule('0 0 * * *');
```

For more information about the format of the schedule, see [UNIX cron format](#).

- For example, for SQL Server:

```
EXECUTE "IS_Public".Set_Purge_Soft_Delete_Schedule  
@freq_type=4,@freq_interval=1,  
@freq_subday_type=1,@active_start_time = 00000;
```

For more information about the arguments and values for the schedule, see [sp_add_jobschedule](#).

Note: You must not specify values for the @job_id, @job_name, @name, @enabled arguments. i2 Analyze provides the values for these arguments.

In this example, the Information Store is configured to create the jobs that purge any soft-deleted records every day, at midnight.

To turn off automatic purging, call the IS_Public.Remove_Purge_Soft_Delete_Schedule stored procedure.

Regardless of whether a purge was started manually or automatically, the effect is always the same: a set of jobs is created to remove soft-deleted records from the Information Store. To inspect the status and outcome of those jobs, you can use a view that the Information Store provides:

- In IBM Data Studio or SQL Server Management Studio, open the IS_Public.Deletion_By_Rule_Log view and look for blocks of jobs where the rule_name is PURGE_SOFT_DELETE.

For more information about the contents of the IS_Public.Deletion_By_Rule_Log view, see [“Deletion_By_Rule_Log view” on page 450](#).

Deleting records by rule

If you need to delete records from the Information Store for policy reasons, rather than to reflect delete operations that happened elsewhere, you can use the deletion-by-rule mechanism. You can directly delete any records that you select by specifying their property values and metadata.

Through deletion by rule, you can target records that match specific conditions and automate deletion to occur regularly. Deletion can be applied consistently across all of the records in the Information Store, whether ingested from an external source or created in i2 Analyst's Notebook Premium. You can delete records by their source, their update time, or by any of their property values.

A deletion-by-rule operation has the following components:

- A condition specifies the particular records in the Information Store that you want to delete.
- A rule contains the condition in a form that you can store, together with a name and information about how you want a deletion job to be created.
- A job contains the rule and is placed in a queue to run deletion on the specified records.

For deleting records by rule, IBM i2 Analyze provides the following procedures and views:

- Deletion views are based on the entities and links that are defined in the i2 Analyze schema. These views are a simplified representation of the underlying data structures for composing conditions.
- Procedures are supplied for defining and managing deletion rules. If you want a job to be scheduled to run automatically, set the rule to be automated.
- Procedures are supplied for creating deletion jobs based on existing rules.
- Views provide a list of deletion rules, mappings between views and schema types, and a log of deletion jobs, status, and outcomes.

The procedures and views that are associated with deletion by rule are generated automatically when you deploy an Information Store or when you upgrade from an earlier version.

You can construct an SQL query that identifies the data that is to be deleted from the Information Store. When you verify that the correct data is returned in response to the SQL query, you can use that condition to create a rule as a basis for a deletion job. You do not need to have a detailed understanding of the database structure. You do need a knowledge of the i2 Analyze schema. For more information, see the i2 Analyze [data model](#) documentation.

When you want to run a deletion job only once or irregularly, you can create it on a manual basis. You might also have deletion jobs that you want run regularly on automated basis. All automated jobs are created according to the same schedule and placed in a queue of jobs to run in order of their creation time.

Running deletion-by-rule jobs on an automated basis requires the Db2 administrative task scheduler if you are using Db2, or the SQL Server Agent to be running if you are using SQL Server. For more information, see the [i2 Analyze software prerequisites](#). You can configure the schedule for deletion-by-rule job creation according to your requirements.

Deleting data by rule

Use an SQL query to identify the data to be deleted, then create and store the deletion rule to identify the corresponding data records for deletion. You can define the rule to specify deletion on an automated schedule or a manual basis.

If you want to apply deletion by rule to a remote Information Store hosted on a Db2, you must catalog the remote database by using the local IBM Db2 client. For more information, see the first note in [Configuring remote IBM Db2 database storage](#).

By default, deletion-by-rule privileges are granted to the Information Store database administration user. If necessary, the privileges can be granted to other users by giving them the `Deletion_By_Rule` role. For more information, see [Authorization to delete by rule](#). The privileges give you access to all the deletion-by-rule views and procedures, as summarized in the following outline of the main tasks.



CAUTION: Grant the `Deletion_By_Rule` role only to users with sufficient knowledge and authority. Use caution when you complete deletion-by-rule tasks as they constitute a powerful mechanism for deletion of data that might be difficult to recover. Ensure that there is a reliable backup in place before you delete by rule.

1. Identify the data to be deleted by composing and running an SQL query to select the data from the deletion view.
You can use a different view to see details of the deletion views for your database.
2. Check the SQL results to confirm that the selected data is as you expected.
You can use the condition in subsequent steps to identify the particular records in the Information Store that you want to be deleted.
3. Create a deletion rule based on the SQL query you verified at step 2 by using the supplied stored procedure.
By default when a rule is created, automated job creation is switched off. You can use another view to see a list of rules and their automation status.
4. Create a deletion job manually by using the supplied stored procedure.
The procedure creates a deletion job that contains the rule, which is queued to run.
5. Check the status of the job, and that deletion occurred as you expected, by consulting the `IS_Public.Deletion_By_Rule_Log` view.
For more information, see [Verifying deletion by rule](#).
6. Optional: Automate deletion by rule by using the supplied stored procedure to include the rule in automated job creation.

Identifying the data

How you compose a rule to delete data from the Information Store varies according to the structure and content of your database. The first step is to understand what you need to delete, and then create an SQL query that selects that data from the deletion views.

Use the deletion views as a basis to create and validate an SQL query that selects the data you want to delete. For more information, see [“Deletion views and columns”](#) on page 446.

You can complete this task in IBM Data Studio or Microsoft SQL Server Management Studio.

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. By using the data in a deletion view, identify the criteria that you can use to delete specific data.
3. Create your SQL query.

For example, based on the deletion view `IS_Public.E_Event_DV`, create a query that returns all event entities where the last update in the source data was more than three years ago:

- For Db2:

```
SELECT *
FROM IS_Public.E_Event_DV
WHERE source_last_updated < CURRENT_DATE -3 YEARS
```

- For SQL Server:

```
SELECT *
FROM "IS_Public".E_Event_DV
WHERE source_last_updated < DATE_ADD(yyyy, -3, GETDATE())
```

4. Verify that the data that is returned from your SQL query is the data that you want to delete.

Create a deletion rule based on your query. For more information, see [Creating a deletion rule](#).

Creating a deletion rule

Use a valid SQL query as a basis for a deletion rule. Store the corresponding deletion rule for use in the deletion process.

Verify the condition that you need for the deletion rule by creating an SQL query that returns the appropriate data. For more information, see [Identifying the data](#).

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. Run the `IS_Public.Create_Deletion_Rule` stored procedure. You must specify the mandatory parameter values for the stored procedure.
 - a) To give the rule a name by which it can be identified, enter a unique name in **Rule_Name**.
Rule names are not case-sensitive.
 - b) To identify the deletion view that the rule targets, enter the name in **View_Name**.
Enter the view name without the `IS.Public` schema name prefix.
 - c) Enter the `WHERE` clause conditions from your SQL query in **Conditions**.
Enter the code that follows the `WHERE` keyword in the SQL query.

For example, see the sample parameter values in the following table. Mandatory values are marked with an asterisk.

Name	Type	Value
Rule_Name*	VARCHAR(50)	OLD EVENTS
View_Name*	VARCHAR(128)	E_Event_DV
Conditions	VARCHAR(1000)	source_last_updated < CURRENT_DATE -3 YEARS

Running this procedure does not run the rule. However, the rule is validated to inform you of any errors. If the rule has no conditions, you receive a warning message to ensure that you are aware of the consequences when you do create a deletion job based on the rule



CAUTION: When a job is created from a rule that has no conditions, it deletes every record in the view.

To review the rule, browse the data in the `IS_Public.Deletion_Rules` view. For example, see the following rule details.

rule_name	view_name	conditions	automated
OLD EVENTS	E_Event_DV	source_last_updated < CURRENT_DATE -3 YEARS	N

By default, the rule is set up so that you can use it to create jobs manually. For more information, see [“Creating a deletion job” on page 444](#). You can also configure the rule so that jobs are created automatically, according to a schedule. For more information, see [Automating deletion by rule](#).

You can update a stored rule by a procedure similar to rule creation, or use a procedure to delete a rule. For more information, see [“Stored procedures” on page 448](#).

Creating a deletion job

Creating a rule stores it with a name that you can use to refer to it later. To delete records that match the rule, you must create a deletion job.

Create a deletion rule. For more information, see [Creating a deletion rule](#).

When you create a job, the rule is placed in the job queue and initially has a status of Pending. At 10-second intervals, i2 Analyze checks for deletion jobs to run.

Note: Do not run a deletion job while data is being ingested into the Information Store.

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. To create the deletion job, run the `IS_Public.Create_Deletion_Job` stored procedure and enter the name of the rule.

You can check the status of the job in the `IS_Public.Deletion_By_Rule_Log` view. For more information, see [Verifying deletion by rule](#).

You can automate the rule for deletion on a schedule. For more information, see [“Automating deletion by rule” on page 445](#).

Verifying deletion by rule

You can verify the outcome of each deletion-by-rule job by consulting the `IS_Public.Deletion_By_Rule_Log` view. The status indicates whether an instance of deletion based on a rule is pending, succeeded, or failed.

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. To verify the status of deletion jobs, browse the data in the `IS_Public.Deletion_By_Rule_Log` view.
3. Find the rule and job that you are interested in. You can use the `rule_name` and `job_creation_time` columns to do find the rule.
4. Verify the status or the outcome of the deletion process in the status column.

For more information about the contents of the log, see [“Deletion_By_Rule_Log view” on page 450](#).

When the job status is Succeeded, you can browse the data in the deletion view for indications that the effect of the job on the database is as expected. Alternatively, you can run the SQL statement and verify that there are no matching records.

When the job fails, the job status indicates an error. For more information, see [“Troubleshooting” on page 446](#).

Automating deletion by rule

By default, new deletion rules are configured so that you can create deletion jobs manually. You can configure each rule so that deletion jobs are created automatically according to a schedule.

Running deletion-by-rule jobs on an automated basis requires the Db2 administrative task scheduler if you are using Db2, or the SQL Server Agent to be running if you are using SQL Server. For more information, see the [i2 Analyze software prerequisites](#).

When you automate a deletion rule, it is included in a deletion job that is created by a scheduled task. i2 Analyze processes in sequence all the jobs that are in the queue whether created manually or automatically.

To automate a rule, complete the following steps.

Note: Do not run a deletion job while data is being ingested into the Information Store. For more information about the deletion-by-rule job schedule, see [“Changing the automated job creation schedule” on page 451](#).

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. Run the `IS_Public.Automate_Deletion_Rule` stored procedure and enter the name of the rule.
The automated flag for the rule is changed from N to Y.
3. To view the rule details and confirm an automated setting, browse the data in the `IS_Public.Deletion_Rules` view.

At the time when automated deletion by rule is scheduled, a job is created for each rule that is set to automated. Each job is queued and runs in sequence.

You can check the status of the job in the `IS_Public.Deletion_By_Rule_Log` view. For more information, see [“Verifying deletion by rule” on page 445](#).

You can configure the schedule for deletion-by-rule job creation according to your requirements. For more information, see [“Changing the automated job creation schedule” on page 451](#).

To disable automated deletion for a rule, run the procedure `IS_Public.Disable_Deletion_Rule` and enter the rule name.

Troubleshooting

When a deletion by rule job is completed, information that indicates success or failure is sent to the `IS_Public.Deletion_By_Rule_Log` view. More details are recorded in the `IS_Public.Ingestion_Deletion_Reports` view.

You can see the status of all deletion-by-rule jobs in the `IS_Public.Deletion_By_Rule_Log` view. For more information, see [“Verifying deletion by rule” on page 445](#).

If the value of `status` is not `Succeeded`, the type of error that is recorded informs your diagnosis and troubleshooting steps. The possible status values, explanations, and steps to take for each value are described as follows.

Pending: status unchanged after a significant amount of time

1. Check other jobs in the `IS_Public.Deletion_By_Rule_Log` view to see whether there is a backlog due to the number and complexity of jobs; or jobs are not being processed as expected and a significant amount of time has elapsed since the most recent job was processed.
2. Check whether the server is running. You might need to restart the server. For more information, see [Deploying i2 Analyze](#).
3. Check whether the server is connected to the database. There might be a network issue. You can find the Opal server logs here: `IBM\i2analyze\deploy\wlp\usr\servers\opal-server\logs`

If you are applying deletion by rule to a remote Information Store on Db2, the remote database must be cataloged by using the local Db2 client. If the configuration files are not already updated, complete the update by using the toolkit. For more information, see [Configuring remote IBM Db2 database storage](#).

If you are using IBM Db2 and none of the previous steps resolve the issue, there might be a problem with the Administrative Task Scheduler or Db2. For more information, see [Troubleshooting administrative task scheduler](#).

Server error

1. To assess the extent of the problem, in the `IS_Public.Deletion_By_Rule_Log` view, check the `reject_count` value to see the number of items that are rejected.
2. Access the `IS_Public.Ingestion_Deletion_Reports` view for more information. A specific deletion job can be identified by the `job_id` value, which is identical to the `job_id` value in the `IS_Public.Deletion_By_Rule_Log` view.

Note: Deletion-by-rule automatically populates the `label` column in the `IS_Public.Ingestion_Deletion_Reports` view with the value `Deletion rule:<Deletion Rule Name>`.

Deletion views and columns

Deploying i2 Analyze automatically generates a set of deletion views in the `IS_Public` schema that is part of the Information Store. Using the views, you can browse the record data and write SQL queries that select records for deletion.

Use the `IS_Public.Deletion_View_Lookup` view to determine the deletion view names in your Information Store. You can also use the view to understand how the deletion views relate to the schema type IDs. The following table contains some examples.

Table 7. Extract from a sample Deletion_View_Lookup view

Schema type identifier	Display name	Deletion view name
ET2	Event	E_Event_DV
ET5	Person	E_Person_DV
ET8	Communications device	E_Communications_Device_DV
ET10	Account	E_Account_DV
LCO1	Communication	L_Communication_DV
LAS1	Associate	L_Associate_DV
LAC1	Access To	L_Access_To_DV

There is a deletion view for each record type. Each of these views has a suffix of `_DV`. Prefixes of `E_` and `L_` are used for entities and links. In IBM Data Studio or SQL Server Management Studio, you can expand a view to see its columns.

Any columns with a prefix of `P_` contain values of properties that are defined in the `i2 Analyze` schema. The other columns contain metadata values that it might be useful to base rules on. Each view has columns for property values, which vary by item type, and columns for metadata values that are common to all records. The following table contains some examples of property columns.

Table 8. Sample extract of deletion views and properties from an IS_Public schema

Deletion view name	Column name	Sample value
E_Person_DV	p_unique_reference	P86539K
	p_date_of_birth	1932-10-14
	p_gender	Male

L_Access_To_DV	p_unique_reference	ADC153
	p_type_of_use	Account Holder

Metadata columns can be useful for creating deletion rule conditions. All deletion views have a common set of metadata columns as described in the following table.

Table 9. Metadata columns that are common across deletion views

Column name	Description
record_id	The record identifier that distinguishes the <code>i2 Analyze</code> record uniquely throughout the deployment
item_id	The item identifier that distinguishes <code>i2 Analyze</code> records of a particular type within the Information Store
last_update_time	The timestamp of the most recent <code>i2 Analyze</code> data ingestion
create_time	The timestamp that <code>i2 Analyze</code> assigned at the point of creation of the record
source_names	The names of the sources of the data

Column name	Description
source_created	The timestamp from the external source for the creation of the data
source_last_updated	The timestamp from the external source for the last update of the data

Note:

- In the Information Store, there is a one-to-one mapping between record identifiers and item identifiers. For more information about the identifiers that are used in i2 Analyze, see [Identifiers in i2 Analyze records](#).
- Records that are created through IBM i2 Analyst's Notebook have a value of ANALYST in the source_names column and null values for source_created and source_last_updated.

Views that represent link types have a few extra metadata columns as described in the following table.

Column name	Description
from_record_id, from_item_id	The identifiers of the record that constitutes the start of a link
to_record_id, to_item_id	The identifiers of the record that constitutes the end of a link
from_item_type_id	The identifier of the type of the record that constitutes the start of a link
to_item_type_id	The identifier of the type of the record that constitutes the end of a link

Each deletion view has a security dimension column, which contains the security dimension values for each record. For more information, see [Security model](#). It can be useful to delete records based on their security dimension values.

Note: The deletion of data is not subject to restriction based on the i2 Analyze security dimension values, but you can filter data for deletion based on these values.


Stored procedures

The i2 Analyze deletion-by-rule functions depend upon a set of standard stored procedures. You can use these procedures to manage deletion rules and create deletion jobs.

The procedures that are available are described in the following table. An asterisk indicates a mandatory parameter.

Procedure	Parameters	Description
IS_Public.Automate_Deletion_Rule	RULE_NAME*	Set automated to Y for the named rule to be included in the scheduled automatic creation of deletion jobs. Deletion-by-rule jobs are placed in the queue of deletion jobs to be run in sequence.

Table 11. Deletion by rule: stored procedures (continued)

Procedure	Parameters	Description
IS_Public.Create_Deletion_Rule	RULE_NAME* VIEW_NAME* CONDITIONS	Convert the SQL conditions and named view to a rule that is ready for deletion job creation.  CAUTION: When a job is created from a rule that has no conditions, it deletes every record in the view.
IS_Public.Delete_Deletion_Rule	RULE_NAME*	Delete the named rule.
IS_Public.Disable_Deletion_Rule	RULE_NAME*	Set the automated parameter to N for the rule to be excluded from the scheduled automatic creation of deletion jobs. The rule can be used for manual deletion only.
IS_Public.Create_Deletion_Job	RULE_NAME*	Create a deletion job that is based on the named rule. This procedure can be run for the rule when the value of automated is Y or N. The deletion-by-rule job is placed in the queue of deletion jobs to be run.
IS_Public.Update_Deletion_Rule	RULE_NAME* VIEW_NAME* CONDITIONS	Update the named rule by revising the original VIEW_NAME or CONDITIONS .
IS_Public.Update_Deletion_Schedule	SCHEDULE*	Update the job creation schedule. For more information about the schedule format, see “Changing the automated job creation schedule” on page 451.
IS_Public.Validate_Deletion_Rule	RULE_NAME*	Report on the SQL Results tab any SQL errors that are found associated with the SQL conditions of the named rule. The report contains results by MESSAGE , SQLCODE , and SQLSTATE values.

Deletion_By_Rule_Log view

The `IS_Public.Deletion_By_Rule_Log` view provides information on the status of the deletion jobs in the queue in relation to each rule. The view also contains details on the results of each deletion job that is run.

You can access the view to check the status of a deletion-by-rule job. For more information, see [Verifying deletion by rule](#). After the deletion job is run, the `IS_Public.Deletion_By_Rule_Log` view contains one or more entries per job that describe the result for each item type. For more information, see [“Sample use cases” on page 452](#).

The information that is contained in the `IS_Public.Deletion_By_Rule_Log` view is described in the following table.

Column name	Description
<code>status</code>	Status of the deletion by rule job: <ul style="list-style-type: none">• Succeeded• Pending• Server error• Validation error
<code>rule_name</code>	Name of the deletion rule.
<code>view_name</code>	Name of the view that the rule is targeting.
<code>conditions</code>	Deletion conditions that are applied by the rule.
<code>job_id</code>	Identifier for the job, which is also used to identify the job in the <code>job_id</code> column of the <code>Ingestion_Deletion_Reports</code> view.
<code>job_creation_time</code>	Time stamp that indicates when the deletion by rule job was created.
<code>job_run_time</code>	Time stamp that indicates when the deletion by rule job was run and corresponds to the value in the <code>start_time_stamp</code> column of the <code>Ingestion_Deletion_Reports</code> view. There is no value in the <code>job_run_time</code> column while the job is pending.
<code>item_type</code>	Item type of the records that were deleted. There can be more than one entry in this view for the same job. For example, deleting entity records can also cause link records to be deleted.
<code>delete_count</code>	The number of records that are deleted.
<code>reject_count</code>	The number of records that are rejected.

Example of job results logged

See the following tables for an example of how the `IS_Public.Deletion_By_Rule_Log` view might look with completed job details.

status	rule_name	view_name	conditions
Succeeded	OLD ACCOUNTS	E_Account_DV	p_date_opened < CURRENT_DATE - 50 YEARS
Succeeded	OLD ACCOUNTS	E_Account_DV	p_date_opened < CURRENT_DATE - 50 YEARS

job_id	job_creation_time	job_run_time	item_type	delete_count	reject_count
77	2017-12-12 14:29:11.351	2017-12-12 14:29:16.467	LAC1	3	0
77	2017-12-12 14:29:11.351	2017-12-12 14:29:16.467	LTR1	3	0

In this example, deleting three Account entity records caused the deletion in the same job of three Transaction link records that were attached to the Accounts.

Changing the automated job creation schedule

i2 Analyze creates jobs from all automated deletion rules according to the same schedule. By default, it creates deletion-by-rule jobs every hour, on the hour. You can update that schedule by running a stored procedure.

Depending on the database management system that hosts the Information Store database, the IBM Db2 administrative task scheduler or Microsoft SQL Server Agent runs the task that creates deletion-by-rule jobs at intervals that you can configure.

The setting that controls how often the deletion-by-rule jobs are created is stored in the Information Store. Whenever the setting matches the current time, date, and day, the scheduler creates jobs from the deletion rules that have the value of **automated** set to Y.

When you deploy i2 Analyze, the setting receives its default value that means that the scheduler looks for jobs to create every hour, on the hour.

- For Db2, a value of '0 * * * *' is used.

For more information about the format, see [UNIX cron format](#).

- For SQL Server, the following arguments and values are used:

@freq_type=4,@freq_interval=1,@freq_subday_type=8,@freq_subday_interval=1

For more information about the arguments and values, see [sp_add_jobschedule](#).

Note: You must not specify values for the @job_id, @job_name, @name, @enabled arguments. i2 Analyze provides the values for these arguments.

To change the schedule for automated deletion, run the IS_Public.Update_Deletion_Schedule stored procedure. For example, to run the deletion-by-rule job creation task only at midnight, run the stored procedure in the following ways:

If you are using Db2, you must specify a UNIX cron value for the schedule. For example, run the stored procedure with the following value:

```
CALL IS_Public.Update_Deletion_Schedule('0 0 * * *');
```

Note: The Db2 administrative task scheduler checks for new or updated tasks at 5-minute intervals. If you change the setting, be aware of a potential delay of 5 minutes before it takes effect. You can still create a job from the rule manually if you need to.

If you are using SQL Server, you must specify arguments for the `sp_add_jobschedule` stored procedure. For example, run the stored procedure with the following value:

```
EXECUTE "IS_Public".Update_Deletion_Schedule  
    @freq_type=4,@freq_interval=1,  
    @freq_subday_type=8,@freq_subday_interval=24;
```

Note: If you are using Db2 and you follow this procedure in a deployment that provides high availability, you must run the stored procedure on the primary and any standby database instances.

Authorization to delete by rule

The deletion-by-rule capability is enabled by the i2 Analyze Information Store deployment. By default the database user who deploys the Information Store is authorized to delete by rule.

Authorization to delete by rule includes permission to access and run all of the views and procedures for doing so in the `IS_Public` schema. For more information, see [“Deletion views and columns” on page 446](#) and [“Stored procedures” on page 448](#). If you do not have this authorization, you can see but not access or run these views and procedures.

The authorization to delete by rule can be granted to other users. You can create a new database user and assign the `Deletion_By_Rule` role to that user, or you can assign the role to an existing user.

For example, you can run the following statement for Db2:

```
GRANT ROLE Deletion_By_Rule TO USER deletion_user
```

You can run the following statement for SQL Server:

```
ALTER SERVER ROLE Deletion_By_Rule ADD MEMBER deletion_user
```



CAUTION: Assign the `Deletion_By_Rule` role only to users with sufficient knowledge and authority. Exercise caution when you create deletion rules as they constitute a powerful mechanism for deletion of data that cannot be recovered afterward.

Sample use cases

Stored data might be monitored for compliance with international standards, corporate regulations, or storage volume restrictions. Depending on the use case, different rules can be created to meet many different kinds of requirement.

Reasons for data deletion often fall into one or more of the following categories:

- Data is out of date
- Data is obsolete or irrelevant in the context of the purpose of the database
- Data has data privacy or security issues
- Data reaches a storage volume threshold

The SQL statements in the following examples contain the kinds of conditions that are commonly used as a basis for deletion.

Note: If you are using Db2, when you are creating SQL for time-based conditions you can use the following documentation on date and time arithmetic support. For more information, see [Datetime operations and durations](#).

Selecting by property value

Select records of all persons who are below a specific age. The rule matches records of persons who are younger than 18 years. This example is one that you might use to delete data that does not suit the specific purpose of the database.

- For Db2:

```
SELECT *
FROM IS_Public.E_Person_DV
WHERE p_date_of_birth > CURRENT_DATE - 18 years
```

- For SQL Server:

```
SELECT *
FROM "IS_Public".E_Person_DV
WHERE p_date_of_birth > DATEADD(yyyy, -18, GETDATE())
```

Metadata: data source

Select all records that are sourced from the "DMV" data source.

```
SELECT *
FROM IS_Public.E_Person_DV
WHERE source_names LIKE '%DMV%'
```

Metadata: creation date

Use SQL to select any record with a creation date that is not in the current calendar year.

- For Db2:

```
SELECT *
FROM IS_Public.E_Person_DV
WHERE YEAR(create_time) < YEAR(CURRENT_DATE)
```

- For SQL Server:

```
SELECT *
FROM IS_Public.E_Person_DV
WHERE create_time < DATEFROMPARTS(DATEPART(yyyy, GETDATE()), 1, 1)
```

Metadata: old data

Select communication records older than three months.

- For Db2:

```

SELECT *
FROM IS_Public.L_Communication_DV
WHERE create_time < CURRENT_DATE - 3 MONTHS

```

- For SQL Server:

```

SELECT *
FROM IS_Public.L_Communication_DV
WHERE create_time < DATEADD(mm, -3, GETDATE())

```

Without the specified link

This case is an example where you require deletion of entities that are not linked to other entities. You might want this type of deletion as a general database cleanup, or to target links of a specific type, for example those granting access to a building or an account. Select records of people who do not have any access links.

The rule matches records of people when the `item_id` is not contained in a `from_item_id` column in the link deletion view. When your rule joins to another deletion view, it is recommended that you identify records by using the `item_id` columns.

```

SELECT *
FROM IS_Public.E_Person_DV
WHERE item_id NOT IN (
    SELECT from_item_id
    FROM IS_Public.L_Access_To_DV
)

```

With the specified links

Select records of any person who is associated with a person named 'Michael Wilson'.

When your rule joins to another deletion view, it is recommended that you identify records by using the `item_id` columns.

```

SELECT *
FROM IS_Public.E_Person_DV
WHERE item_id IN (
    SELECT A.from_item_id
    FROM IS_Public.L_Associate_DV AS A
    INNER JOIN IS_Public.E_Person_DV AS P
        ON P.item_id = A.to_item_id
    WHERE P.p_first_given_name = 'Michael' AND
        P.p_family_name = 'Wilson'
)

```

Data source, property, and timeframe

This case is an example of the kind of deletion that might be used to clean up a database or reduce storage requirements. Select records from a specific source based on when the last update occurred. The rule matches records of people with the named data source and an update date that is older than three years from the current date.

- For Db2:

```
SELECT *
FROM IS_Public.E_Person_DV
WHERE source_names LIKE '%LawEnforcementDB%' AND
source_last_updated < CURRENT_DATE - 3 YEARS
```

- For SQL Server:

```
SELECT *
FROM IS_Public.E_Person_DV
WHERE source_names LIKE '%LawEnforcementDB%' AND
source_last_updated < DATEADD(yyyy, -3, GETDATE())
```

Retrieving chart metadata

i2 Analyze enables database administrators and third-party extensions to retrieve information about charts that users upload to the Information Store or the Chart Store. Specifically, you can find out which users have uploaded a particular chart, and when that chart was last accessed.

Note: In this context, "accessed" refers to when the chart was uploaded, downloaded, or had its details (that is, its history) requested by a user.

With this knowledge, you might implement a policy such that charts that haven't been accessed for a certain period of time are automatically deleted - but only after you send an alert to all the users who uploaded the chart to warn them.

Determining when a chart was last accessed

To determine when a chart was last accessed, you can query the chart deletion view, whose name has the following format: `IS_Public.<ChartTableName>_DV`. Provided that the item type in the i2 Analyze schema that represents charts has not been modified, its name is `IS_Public.E_Analyst_S_Notebook_Ch_DV`.

This view has the same structure as [other deletion views](#), but with two extra columns:

Column name	Description
chart_type	The type of the chart - that is, web or desktop
last_seen_time	The (UTC) date and time when the chart was last accessed

For this purpose, the `last_seen_time` column is pertinent. To determine which charts have not been accessed for approximately 12 months, you might execute the following (Db2) SQL statement:

```
SELECT item_id
FROM IS_Public.E_Analyst_S_Notebook_Ch_DV
WHERE last_seen_time < NOW() - 12 MONTHS;
```

You can then use the list of identifiers that this statement returns to find out which users need to be notified.

Determining which users uploaded a chart

The Information Store database provides a view that contains information about which users have uploaded a version of a particular chart. The default name of this view is `IS_Public.E_Analyst_S_Notebook_Ch_Upload_Users`. It has the following columns:

Column name	Description
<code>item_id</code>	The item identifier of the chart record
<code>user_principal_name</code>	The name of a user who uploaded any version of the chart

Therefore, to determine which users have uploaded (any version of) a chart that has an `item_id` of 1234, you might execute the following SQL statement:

```
SELECT user_principal_name
FROM IS_Public.E_Analyst_S_Notebook_Ch_Upload_Users
WHERE item_id = 1234;
```

Note: Even if the same user has uploaded multiple versions of the same chart, their principal name is only returned once for each chart.

With this list of user names in hand, you might decide to use one of the i2 Analyze APIs for [sending alerts to users](#).

Sending alerts to i2 Analyze users

i2 Analyze enables system administrators and third-party extensions to send alerts to users of the system. These alerts appear in the Analyst's Notebook Premium client alongside the alerts from Visual Queries that users are familiar with.

In principle, alerts can contain any information that you think it might be useful for i2 Analyze users to receive. In practice, alerts are often about data in the system - to inform users about a change, perhaps, or to warn them that something temporary is soon to be deleted.

Important: Alerting is always enabled in deployments that include the Information Store, but by default it is *not* enabled in deployments that include only the Chart Store. To enable alerting in a Chart Store deployment, you must set `EnableAlerting=true` in `DiscoServerSettingsCommon.properties`, and then restart the i2 Analyze server.

Creating alerts

The alerts that you create contain an icon, a title, and an optional message. In addition, they can contain either a group of records or a link to an external resource. They can be created through three different APIs:

- [Database stored procedures](#)
- [A Java API](#)
- [A REST API](#)

Each API provides the same control over the contents of alerts. The Java and REST APIs provide additional functionality for specifying the recipients of alerts, based on group membership or command access control permissions. (The stored procedures require recipients to be specified explicitly, one at a time.)

Note: Alerts can only be sent to users who have previously logged in to i2 Analyze. An authorized user who has never logged in will not see alerts that were created and sent before they do so.

Alert structure

The structure of an alert is the same no matter which API creates it. Alerts can contain the following elements:

- **title**

The title is mandatory and should contain a short string that will be displayed for the alert.

- **message**

The message is optional. If supplied, it can contain a longer string with more information than the title.

- **icon**

The icon is optional and determines the image that is shown next to the alert in the user interface. If supplied, it must be one of the following values. If not supplied, it defaults to INFORMATION.

- SUCCESS
- INFORMATION
- WARNING
- ERROR
- PLUS
- MINUS

- **record identifiers**

Record identifiers are optional. If they are supplied, the alert links to the specified records or charts and displays them in a results grid (in the same way that Visual Query alert results are displayed).

Record identifiers cannot be supplied if an href is supplied.

- **href**

The href is optional. If supplied, it should contain a URL for a resource that provides more information or context for the alert. The URL is displayed as a hyperlink in the user interface.

An href cannot be supplied if record identifiers are supplied.

Database API

The Information Store (or Chart Store, when alerting is enabled) database contains two public stored procedures for creating alerts. To create an alert with an optional hyperlink, use `IS_Public.Create_Alert`, which takes the following parameters:

Parameter	Alert element	Db2 type	SQL Server type	Size
user_principal_name	n/a	VARCHAR	NVARCHAR	256

Parameter	Alert element	Db2 type	SQL Server type	Size
alert_title	title	VARGRAPHIC	NVARCHAR	100
alert_message	message	VARGRAPHIC	NVARCHAR	1000
alert_link	href	VARGRAPHIC	VARCHAR	256
alert_icon	icon	VARCHAR	VARCHAR	20

For example:

```
[Db2]
CALL IS_Public.Create_Alert('test-user', 'For your information', 'Take a look at this link',
'https://example.domain/news/123', 'INFORMATION')

[SQL-Server]
EXEC IS_Public.Create_Alert 'test-user', 'For your information', 'Take a look at this link',
'https://example.domain/news/123', 'INFORMATION'
```

Note: To send the same alert to multiple users, you must execute the stored procedure multiple times, changing the user_principal_name as required.

To create an alert with one or more record identifiers, use IS_Public.Create_Records_Alert, which takes the following parameters:

Parameter	Alert element	Db2 type	SQL Server type	Size
user_principal_name	n/a	VARCHAR	NVARCHAR	256
alert_title	title	VARGRAPHIC	NVARCHAR	100
alert_record_ids_csv	record identifiers	VARCHAR	VARCHAR	8000
alert_message	message	VARGRAPHIC	NVARCHAR	1000
alert_icon	icon	VARCHAR	VARCHAR	20

For example:

```
[Db2]
CALL IS_Public.Create_Records_Alert('test-user', 'Records added!',
'h6cBtBz6CuYEU3YwzqUuZEmrhJ,TaUjsCpKUnmY85YWgBQbeNqou2', 'New records added from example source',
'PLUS')

[SQL-Server]
EXEC IS_Public.Create_Records_Alert 'test-user', 'Records added!',
'h6cBtBz6CuYEU3YwzqUuZEmrhJ,TaUjsCpKUnmY85YWgBQbeNqou2', 'New records added from example source',
'PLUS'
```


Note: Alerts that you create through this stored procedure can contain at most 242 records, because of the sizes of record identifiers and the `alert_record_ids_csv` parameter. The Java and REST APIs call the stored procedure, so they have the same limit.

Java API

The Java API for creating alerts is available through the API for creating custom, scheduled tasks.

The API provides the ability to identify the recipients of an alert by specifying lists of users, groups, and command access control permissions. The criteria are ORed together inclusively, so only one criterion must be met in order for a user to receive an alert. It is also possible to specify that an alert should be sent to all system users.

Note: You can send an alert to all users that are in the Analyst group OR the Clerk group. But you *cannot* specify that an alert should be sent only to users that are members of both the Analyst group AND the Clerk group.

For example:

```
package com.example.tasks;

import com.i2group.disco.alerts.AlertIcon;
import com.i2group.disco.alerts.IAlertManager;
import com.i2group.disco.alerts.ISendTo;
import com.i2group.disco.alerts.SendTo;
import com.i2group.disco.task.spi.CustomTaskFailedException;
import com.i2group.disco.task.spi.IScheduledTask;
import com.i2group.tracing.ITracer;
import java.sql.SQLException;
import java.util.List;

public class ExampleCustomTask implements IScheduledTask {
    private IScheduledTaskObjects scheduledTaskObjects;

    @Override
    public void onStartup(IScheduledTaskObjects scheduledTaskObjects) {
        this.scheduledTaskObjects = scheduledTaskObjects;
    }

    @Override
    public void run() {
        final IAlertManager alertManager = scheduledTaskObjects.getAlertManager();

        try {
            // Create an alert with a hyperlink
            final ISendTo sendToTestUser = SendTo.createWithUsers("test-user");
            alertManager.createAlert(
                sendToTestUser, "For your information",
                AlertIcon.INFORMATION, "Take a look at this link",
                "https://example.domain/news/123");

            // Create an alert that contains records
            final ISendTo sendToAnalysts = SendTo.createWithGroups("Analyst");
            alertManager.createRecordsAlert(
                sendToAnalysts, "Records added!",
                AlertIcon.PLUS, "New records added from example source",
                List.of("h6cBtBz6CuYEU3YwzqUuZEmrhJ", "TaUjsCpKUnmY85YWgBQbeNqou2"));
        } catch (SQLException ex) {
            final ITracer tracer = scheduledTaskObjects.getTracerFactory().getTracer(getClass());
            tracer.warn("Could not create alert", ex);
            throw CustomTaskFailedException.reschedule("Failed to create alerts. Will try again
later.");
        }
    }
}
```

REST API

The REST API for creating alerts maps to the Java API and provides the same functionality. You can view the documentation for the REST API on a running i2 Analyze server at `/opal/doc/#!/alerts`.

Important: In order to use the REST API for creating alerts, you must be logged in to the i2 Analyze server as a user with the `i2:AlertsCreate` command access control permission.

For example, the following POST request creates an alert that contains a link, and sends it to the user named `test-user`:

```
POST http://<server-name>:9082/opal/api/v1/alerts
{
  "title": "For your information",
  "sendTo": {
    "users": ["test-user"]
  },
  "icon": "INFORMATION",
  "message": "Take a look at this link",
  "href": "https://example.domain/news/123"
}
```

And the next POST request creates an alert that contains records, and sends it to all the users in the `Analyst` group:

```
POST http://<server-name>:9082/opal/api/v1/alerts
{
  "title": "Records added!",
  "sendTo": {
    "groups": ["Analyst"]
  },
  "icon": "PLUS",
  "message": "New records added from example source",
  "recordIdentifiers": [
    "h6cBtBz6CuYEU3YwzqUuZEmrhJ",
    "TaUjsCpKUnmY85YWgBQbeNqou2"
  ]
}
```

In either case, the response might look like this:

```
{
  "sentToCount" : 1
}
```

The `sentToCount` field indicates how many users the alert was sent to. If the count is different from what you expected - too high or too low - you can enable `DEBUG` logging on the `com.i2group.disco.alerts.SendToResolver` class, and send the request again to see an explanation in the server logs of who the alert was sent to.

Deleting alerts

To delete an alert programmatically, use the database view named `IS_Public.Alerts_DV`. Alerts can be deleted through this view with standard SQL expressions.

For example, to delete all alerts, execute:

```
DELETE FROM IS_Public.Alerts_DV;
```

To delete specific alerts, use a `WHERE` clause. For example, to delete all alerts sent more than 30 days ago (on Db2):

```
DELETE FROM IS_Public.Alerts_DV WHERE create_time_stamp < NOW() - 30 DAYS;
```

Note: A user who has received an alert can delete it for themselves through the user interface in i2 Analyst's Notebook Premium.

Scheduling custom server tasks

i2 Analyze supports the development of custom 'tasks' that the server executes on a user-defined schedule. A task is simply a Java class that implements a specific interface and performs a custom action when invoked.

Creating a custom task

The task class must implement the `IScheduledTask` interface, which is provided by the `disco-api.jar` file in the deployment toolkit.

The `IScheduledTask` interface defines the following methods:

- `run()`

The `run()` method contains the code that performs the custom action. This method is called periodically, based on the schedule specified in the deployment properties file.

- `onStartup()`

The `onStartup()` method contains any necessary start-up or initialization code. It also receives an `IScheduledTaskObjects` parameter that provides access to other i2 Analyze APIs that can be used inside the running task. For example, the scheduler tracer can be used for logging, and the [alert manager](#) can be used to send alerts.

This method is called whenever the server initializes an instance of a task, such as when it starts.

- `close()`

The `close()` method contains any necessary tear-down or finalization code. This method is called whenever the server removes an instance of a task, such as when it stops.

Note: Writing implementations of the `onStartup()` and `close()` methods is optional. They have default implementations that do nothing.

If the custom task fails for any reason, you can control the subsequent behavior by throwing a `CustomTaskFailedException` from the `run()` method. Depending on how the exception is constructed, the server might re-run the task immediately, or when it is next scheduled, or never.

Note: If an exception other than a `CustomTaskFailedException` escapes from the `run()` method, the custom task is canceled and will not run again until the scheduler is reinitialized - when the server is restarted, for example.

Deploying a custom task

To deploy a custom task, the Java class (and any supporting classes) must be packaged in a `.jar` file. You must then copy this `.jar` file to the configuration part of the toolkit. Place it the `WEB-INF/lib` sub-folder of the `opal-services` fragment, in the `toolkit/configuration/fragments` folder.

Note: A new fragment specifically for custom tasks is also an option, provided that it is also added to the `topology.xml` file for the deployment.

Configuration

In order for the i2 Analyze server to discover the custom task, entries that identify it must be present in one of the settings files. A typical location for these entries is the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file.

The new entries have two parts, in the following format:

```
CustomTaskScheduler.<TaskName>.Class=<ImplementationClass>
CustomTaskScheduler.<TaskName>.Expression=<CronExpression>
```

Here, `<TaskName>` is a name that identifies a particular task instance; `<ImplementationClass>` is the fully qualified class name for the implementation of that task; and `<CronExpression>` is a cron schedule expression for when the task should run.

The format of the cron expression is:

```
<minute> <hour> <day-of-month> <month> <day-of-week>
```

For example, a cron expression of `0 0 * * *` results in a task running at midnight every day. See the [UNIX cron format documentation](#) for more information.

The following settings define two different tasks that use two different classes, on two different schedules:

```
CustomTaskScheduler.AlertUsersBeforeDeletingCharts.Class=com.example.alert.ScheduledTask
CustomTaskScheduler.AlertUsersBeforeDeletingCharts.Expression=0 0 * * *

CustomTaskScheduler.TableCleaner.Class=com.example.cleaner.CleanupTask
CustomTaskScheduler.TableCleaner.Expression=0 */4 * * *
```

Deployment

When the `.jar` file is copied into the toolkit, and the `DiscoServerSettingsCommon.properties` file is updated with the custom task class name and the cron schedule, the task is ready for deployment to the i2 Analyze server.

To deploy the custom task to the i2 Analyze server, run `setup -t deployLiberty` to update the configuration with your changes. Then, to set up the task to run according to its defined schedule, run `setup -t startLiberty`.

Monitoring a custom task

The scheduler log stores information about the state of all configured tasks, including when a task was last run, when it will next run, whether it has failed, and so on.

The scheduler log can be found in the `IBM_i2_Scheduler.log` file, inside the i2 Analyze server.

Note: If a custom task is configured to use the scheduler tracer (`IScheduledTaskObjects.getSchedulerTracer()`) in its `onStartup()` method, then anything logged through that object is also written to the `IBM_i2_Scheduler.log` file.

Upgrading i2 Analyze

Different deployments of i2 Analyze can contain different combinations of its component parts. To upgrade a particular deployment of i2 Analyze, you must upgrade the deployment toolkit and then configure and upgrade the deployment itself.

This information relates to the latest release of i2 Analyze.

Troubleshooting and support

[IBM Software Support home page](#)

Upgrade paths

To upgrade a deployment of i2 Analyze, you must first upgrade the deployment toolkit that you are using. You then use this upgraded toolkit to upgrade your deployment. The version of your current deployment determines exactly which path to follow.

- If you are upgrading i2 Analyze version 4.3.1 or later, complete the instructions in [“Upgrading to i2 Analyze 4.3.5”](#) on page 465.
- If you are upgrading from version 4.3.0 or earlier, part of the upgrade includes upgrading Solr to version 8, which requires the Solr index to be recreated as part of the process. This procedure can take a significant amount of time, during which your system is offline. To complete the upgrade with a period of downtime, complete the instructions in [“Upgrading to i2 Analyze 4.3.5”](#) on page 465.

Note: To limit the time that your system is offline, you can upgrade to version 4.3.1 first using the online process, and then upgrade to version 4.3.5. As i2 Analyze 4.3.1 is no longer supported, you must contact IBM support for more information about this path. See [Contacting IBM support](#).

Note: From i2 Analyze version 4.3.3, the Analysis Repository, Intelligence Portal, and Onyx services are no longer supported. Deployments that contain an Analysis Repository should migrate data into another repository.

Software prerequisites

If you are upgrading a deployment of i2 Analyze 4.3.0 or earlier that uses SQL Server for the Information Store database, you must install a later version of the ODBC Driver SQL Server and **sqlcmd** utility. For more information, see [Software prerequisites](#).

As part of the i2 Analyze upgrade process, WebSphere Liberty, Solr, ZooKeeper, and Java are updated. You do not need to download and update these prerequisites before you upgrade an existing deployment.

Upgrading to i2 Analyze 4.3.5

To upgrade a deployment of i2 Analyze, you must first install the latest version of the deployment toolkit. After you install the latest version, you can upgrade your deployment.

Before you upgrade your production deployment, use a pre-production or test environment to verify that you can complete the upgrade process successfully and that you are familiar with the procedure. After you test the upgrade process for your deployment, complete the upgrade in your production environment. If you do not have a pre-production or test environment, you can create one. For more information, see [Creating a production deployment](#).

- If you are upgrading a deployment of i2 Analyze 4.3.0 or earlier that uses SQL Server for the Information Store database, you must install a later version of the ODBC Driver for SQL Server and **sqlcmd** utility. For more information, see [“Software prerequisites”](#) on page 94.

- Ensure that you back up your deployment before you complete an upgrade. For more information about backing up your deployment, see [Backing up a deployment](#).

Depending on the scale and complexity of your data, changes of this nature can take time. You might want to plan your upgrade to take place in a period where there is usually no activity, and back up your system before proceeding.

To upgrade the deployment toolkit to version 4.3.5:

1. Stop i2 Analyze:

- In a single-server topology, run `setup -t stop`.
- In a multiple-server topology, see [Stopping and starting i2 Analyze](#).

2. On each server where the deployment toolkit is installed, make a backup of the i2 Analyze directory. For example, `IBM\i2analyze`.

3. Install the i2 Analyze deployment toolkit.

- a) On each server where the toolkit is installed, remove the `toolkit`, `license`, and `swidtag` directories from the existing installation.
- b) Install the i2 Analyze version 4.3.5 deployment toolkit over your existing toolkit. For more information, see [“Installing i2 Analyze” on page 94](#).

4. For each deployment toolkit in your deployment, copy the `configuration` directory that you backed up in step 2 to the `i2analyze\toolkit` directory of the upgraded deployment toolkit that you installed in step 3.

5. If it is not already present, create and populate the `credentials.properties` file. This file must be stored in the following location: `toolkit\configuration\environment` in each deployment toolkit.

For more information about the file, see [The credentials.properties file](#).

6. If you are upgrading a deployment of i2 Analyze 4.3.3 or earlier that uses SQL Server for the Information Store database, update the JDBC driver in your deployment configuration to the Java 11 version.

a) Update the JDBC driver

For more information about which driver to install, see [Specifying the JDBC driver](#).

After you upgrade the deployment toolkit, you can use it to upgrade the deployment to version 4.3.5:

7. Upgrade and start i2 Analyze according to the instructions in [Upgrading an i2 Analyze deployment](#).

8. If you are using the IBM HTTP Server, restart it.

If your deployment includes the ETL toolkit, you must upgrade the ETL toolkit to version 4.3.5 after you upgrade the rest of the deployment. For more information, see [“Upgrading the ETL toolkit” on page 467](#).

After you upgrade, you might need to update the configuration of your deployment for any new or modified configuration settings. For more information about new and modified configuration settings, see [Configuration and database changes](#).

When you start the server after you upgrade, extra processing of the data in the Information Store is completed after the upgrade. During this processing, you might not be able to ingest, update, and delete data in the Information Store. For more information, see [“Information Store processing after you upgrade i2 Analyze” on page 474](#).

Upgrading the ETL toolkit

If your deployment of i2 Analyze uses the ETL toolkit, you must upgrade the ETL toolkit to version 4.3.5 separately from the rest of the deployment. To upgrade the ETL toolkit, you must remove the existing version and replace it with the one that is deployed with version 4.3.5 of the i2 Analyze toolkit.

You must upgrade your deployment to version 4.3.5 before you can upgrade your ETL toolkit.

1. If you modified the connection properties of your ETL toolkit to connect to a remote instance of Db2, make a backup of the properties file. Navigate to the `etltoolkit\classes` directory of your ETL toolkit, and copy the `Connection.properties` file to a location outside of the `etltoolkit` directory.

You can now remove the previous ETL toolkit.

2. After you upgrade your deployment, deploy the ETL toolkit from the upgraded i2 Analyze toolkit at version 4.3.5.

For more information about deploying the ETL toolkit, see [Deploying the ETL toolkit](#).

3. Update the `Connection.properties` file in the new ETL toolkit with the `db.installation.dir` property and value from the backup `Connection.properties` file from your previous ETL toolkit.

The ETL toolkit is upgraded to version 4.3.5 and ready for use by your ETL logic to modify the Information Store.

Upgrading IBM i2 Analyst's Notebook Premium

To take advantage of some of the latest features of i2 Analyze, users must have the latest version of Analyst's Notebook Premium. To upgrade IBM i2 Analyst's Notebook Premium, you use the Analyst's Notebook Premium Installer.

1. Extract the product files from your downloaded distribution of Analyst's Notebook Premium.
2. Using Windows Explorer, browse to the root of the distribution and run **setup.exe**.
3. Follow the prompts to complete the upgrade.

Note: If your previous deployment of i2 Analyze contained an Analysis Repository, the connection to that repository is no longer available after the upgrade. If you had a Local Analysis Repository, the database is retained to allow you to export the data.

Upgrade resources

Depending on the configuration of your deployment, you might need to complete extra tasks to upgrade your system. The extra tasks might need to be completed before or after you upgrade the system.

Upgrading an i2 Analyze deployment

You can use an upgraded i2 Analyze deployment toolkit to upgrade an existing deployment of i2 Analyze. The new features in later versions are available only after the deployment is upgraded successfully.

Before you begin

Complete the instructions in [Upgrading to i2 Analyze 4.3.5](#) to install the latest version of the deployment toolkit and prepare the environment before you upgrade the deployment.

Note: If your deployment of i2 Analyze uses multiple servers, follow the instructions in [Upgrading an i2 Analyze deployment on multiple servers](#), rather than the procedure in this topic.

Procedure

1. Open a command prompt on the server, and navigate to the `toolkit\scripts` directory of the i2 Analyze toolkit.
2. To upgrade the deployment, run the following command:

```
setup -t upgrade
```

If the `create-databases` attribute in the `topology.xml` file is set to `false`, you must run the scripts to update the database manually. The scripts are created in the `toolkit\scripts\database\db2\InfoStore\generated\upgrade` directory. Run the scripts in ascending alphanumeric order. The user that you run the scripts as must have permission to drop and create database tables and views.

If you are using IBM Db2 v10.5, you must also run the following script to set the temporary tablespace size to 32k: `toolkit\scripts\database\db2\InfoStore\create-ts-for-10-5.db2`.

3. If you see the following message, type 'y' to enable the Chart Store with its default settings as part of the upgrade process:

```
The "Analyst's Notebook Chart" item type is added to the i2 Analyze schema.  
The version of the i2 Analyze schema is incremented to: "2"  
The schema found in: C:\IBM\i2analyze\toolkit\configuration\fragments\common\WEB-INF\classes\  
does not have a chart item type. Would you like the toolkit to add it to your schema? (y/n)
```

Alternatively, if you type 'n', the upgrade process exits without enabling the Chart Store. Instead of accepting the default settings, you can open the Information Store schema in Schema Designer to add the "Analyst's Notebook Chart" item type. You can then customize the item type by adding property types before you complete the upgrade.

4. To complete this part of the upgrade and start the application, run the following command:

```
setup -t start
```

What to do next

After you upgrade and restart i2 Analyze, return to perform the rest of the instructions to finish [upgrading the system](#).

Upgrading an i2 Analyze deployment on multiple servers

To upgrade i2 Analyze in a multiple-server deployment topology, you must run the commands to upgrade and start the components of i2 Analyze on each server.

Before you begin

Complete the instructions in [Upgrading to i2 Analyze 4.3.5](#) to install the latest version of the deployment toolkit on the Liberty, Solr, and ZooKeeper servers, and prepare the environment before you upgrade the deployment.

About this task

To upgrade i2 Analyze in a multiple server deployment topology, you must provide an upgraded configuration to each deployment toolkit. Then, you can run the commands to upgrade the components of i2 Analyze on each server.

As you follow the procedure, it is important to keep track of which server each command runs on, and whether you need to specify the hostname of that server to the command.

Run all toolkit commands from the `toolkit/scripts` directory in the deployment toolkit on the specified server in your environment.

Procedure

1. Upgrade and copy the i2 Analyze configuration.

- a. Upgrade the i2 Analyze configuration:

```
setup -t upgradeConfiguration
```

- b. If you see the following message, type 'y' to enable the Chart Store with its default settings as part of the upgrade process:

```
The "Analyst's Notebook Chart" item type is added to the i2 Analyze schema.  
The version of the i2 Analyze schema is incremented to: "2"  
The schema found in: C:IBM\i2analyze\toolkit\configuration\fragments\common\WEB-INF\classes\  
does not have a chart item type. Would you like the toolkit to add it to your schema? (y/n)
```

Alternatively, if you type 'n', the upgrade process exits without enabling the Chart Store. Instead of accepting the default settings, you can open the Information Store schema in Schema Designer to add the "Analyst's Notebook Chart" item type. You can then customize the item type by adding property types before you complete the upgrade.

- c. Provided that all servers have the same configuration, copy the upgraded `toolkit/configuration` to the `toolkit` directory on each Solr and ZooKeeper server in your environment. Accept any file overwrites.

2. Upgrade the ZooKeeper and Solr components of i2 Analyze.

- a. On each ZooKeeper server, run the following command:

```
setup -t upgradeZookeeper --hostname zookeeper.hostname
```

Where `zookeeper.hostname` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zghost>` element in the `topology.xml` file.

- b. On each Solr server, run the following command:

```
setup -t upgradeSolr --hostname solr.hostname
```

Where `solr.hostname` is the hostname of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

3. Upgrade the Information Store database.

On the Liberty server, run the following command:

```
setup -t upgradeDatabases
```

If the `create-databases` attribute in the `topology.xml` file is set to `false`, you must run the scripts to update the database manually. The scripts are created in the `toolkit\scripts\database\db2\InfoStore\generated\upgrade` directory. Run the scripts in ascending alphanumeric order. The user that you run the scripts as must have permission to drop and create database tables and views.

If you are using IBM Db2 v10.5, you must also run the following script to set the temporary tablespace size to 32k: `toolkit\scripts\database\db2\InfoStore\create-ts-for-10-5.db2`.

4. Start the ZooKeeper component of i2 Analyze.

On each ZooKeeper server, start the ZooKeeper hosts:

```
setup -t startZkHosts --hostname zookeeper.hostname
```

5. Upload the Solr configuration to ZooKeeper.

On the Liberty server, run:

```
setup -t createAndUploadSolrConfig --hostname liberty.hostname
```

6. Start the Solr component of i2 Analyze.

On each Solr server, start Solr:

```
setup -t startSolrNodes --hostname solr.hostname
```

7. Upgrade the Solr collections, and upgrade and start Liberty.

On the Liberty server, run the following commands:

```
setup -t upgradeSolrCollections --hostname liberty.hostname
setup -t upgradeLiberty
setup -t startLiberty
```

What to do next

After you upgrade and start i2 Analyze, return to perform the rest of the instructions to finish upgrading the system.

Configuration and database changes

As a result of an upgrade, you might need to consider configuration or database changes that are new or different compared with previous versions of the software. Depending on the default behavior, you might want to modify the configuration to meet your requirements after you upgrade the deployment.

Version 4.3.5

No changes are introduced at i2 Analyze version 4.3.5 that affect upgrade. However, if your deployment includes an Information Store that runs on IBM Db2, this version includes tooling to improve Visual Query performance with properties that contain multiple-line strings.

Version 4.3.4

The following changes are introduced at i2 Analyze version 4.3.4:

Java install location setting moved

The `java.home.dir` setting in the `environment.properties` file is no longer used. The value that was specified for this setting is now used for the `I2A_COMPONENTS_JAVA_HOME` variable in the `setup.in` file.

For more information about how the location of Java is specified, see [Specifying the Java distribution](#).

Additional setting require for client certificate authentication

If your deployment uses client certificate authentication, you must update the `server.xml` in Liberty with a new mandatory attribute. For more information, see [Configuring i2 Analyze for client certificate authentication](#).

Temporary ingestion table space deprecated in Db2

The `IngestionTempTableSpace` setting in the `InfoStoreNamesDb2.properties` file is superseded by the `InternalStagingTableSpace` setting.

Version 4.3.3

The following changes are introduced at i2 Analyze version 4.3.3:

Record matching

If your existing deployment of i2 Analyze included the i2 Connect gateway but not the Information Store, then the upgrade process modifies the contents of `ApolloServerSettingsMandatory.properties` to reclassify your schema as a *gateway schema*. The effect of this change is to modify the identifiers of the item types in the schema.

If your deployment includes match rules files on the server, the upgrade process automatically updates those files to contain the correct item type identifiers. However, if your users have developed local rules files for Find Matching Records, you must edit those files before they reconnect to the server after an upgrade.

On each workstation, follow the instructions to find the `local-fmr-match-rules.xml` file and use the information in [Match rules syntax](#) to update the item type identifiers in the file.

Search results filtering

If your existing deployment of i2 Analyze included the i2 Connect gateway but not the Information Store, then the upgrade process modifies the contents of `ApolloServerSettingsMandatory.properties` to reclassify your schema as a *gateway schema*. If your existing deployment also used a results configuration file, then you must update that file to reflect the reclassification.

Follow the instructions in [Setting up search results filtering](#) and [Understanding the results configuration file](#) to make the necessary changes to the item type identifiers in the file.

Command access control permissions

A new permission can be added to your command access control file. The new permission grants access to the i2 Notebook web client for the Information Store. For more information, see [Controlling access to features](#).

If you do not update your command access control configuration file to include the new permission, or if command access control is not configured in your deployment, access to the feature is denied to all users.

Liberty log location

The Liberty log files from the i2 Analyze application are now located in the `wlp\usr\servers\opal-server\logs\opal-services` directory. Any logs generated from your previous release remain in their current location. Any new log messages are created in the new files in the new location.

For more information about the log files, see [Deployment log files](#).

New Information Store table space in Db2

The `IS_INTSTG_TS` table space is added to the Information Store database. This table space is used to store the internal staging tables that are created as part of the ingestion process. The `InfoStoreNamesDb2.properties` file contains a new `InternalStagingTableSpace` setting.

Version 4.3.2

No changes are introduced at i2 Analyze version 4.3.2 that affect upgrade.

Version 4.3.1 Fix Pack 1

The following changes are introduced at i2 Analyze version 4.3.1.1:

Command access control permissions

New permissions can be added to your command access control file. The new permissions grant access to upload, delete, and read charts with the Information Store. For more information, see [Controlling access to features](#).

If you do not update your command access control configuration file to include the new permissions, access to the features is denied to all users. If command access control is not configured in your deployment, all authenticated users have access to the new features.

Chart storage

For the Information Store database to store charts, the Information Store schema is updated to include the chart item type. After you upgrade the Information Store database, the following tables are added for chart storage:

- `E_ANALYST_S_NOTEBOOK_CH` - the data table, includes the chart properties
- `E_ANALYST_S_NOTEBOOK_CH_BIN` - the binary data table, includes the binary representation of the chart
- `E_ANALYST_S_NOTEBOOK_CH_TXT` - the text data table, includes the text from the chart that can be searched for

Version 4.3.1

The following changes are introduced at i2 Analyze version 4.3.1:

OpenJDK - Java Development Kit

The JDK that is installed by the i2 Analyze toolkit is now the OpenJDK, instead of the Oracle JDK.

When you upgrade your deployment of i2 Analyze, the Oracle JDK is uninstalled and the OpenJDK is installed.

Deletion view column changes

To incorporate record identifiers, the deletion views are updated to include a new `item_id` column. The content of the existing `record_id` column now contains the record identifier for a record. The `item_id` column contains the item identifier.

If your existing deletion rules use the `record_id` column, you might need to update them to use the `item_id` column. For more information, see [Deleting records by rule](#).

Version 4.3.0

The following changes are introduced at i2 Analyze version 4.3.0:

Database transaction log size increase

The database management system now logs more tables during data movement, which causes an increase to the size of the transaction log. This is more noticeable in deployments that contain large amounts of data.

InfoStoreNames.properties file rename

The `InfoStoreNames.properties` file is renamed to `InfoStoreNamesDb2.properties`.

Version 4.2.1

The following changes are introduced at i2 Analyze version 4.2.1:

Configuration fragments

You can deploy the Information Store and i2 Connect in the same deployment of i2 Analyze. To configure a deployment with this topology, some properties files and settings are in a different location:

- If your deployment contained the `opal-services-daod` fragment, it is renamed to `opal-services`. The settings in the `OpalServerSettingsDaodMandatory.properties` file are now in the `DiscoServerSettingsCommon.properties` file.
- If your deployment contained the `opal-services-is` fragment, the `opal-services` fragment is added and the following files are moved to the `opal-services` fragment:
 - `DiscoClientSettings.properties`
 - `DiscoServerSettingsCommon.properties`
 - Results configuration file
 - `visual-query-configuration.xml`

By default, all of the values for the settings in the files are unchanged.

Command access control permissions

New permissions can be added to your command access control file. The new permissions grant access to the export search results to a CSV file and i2 Connect features in Analyst's Notebook Premium. For more information, see [Controlling access to features](#).

If you do not update your command access control configuration file to include the new permissions, access to the features is denied to all users. If command access control is not configured in your deployment, all authenticated users have access to the features except the export to CSV file feature.

Oracle Java Development Kit

The JDK that is installed by the i2 Analyze toolkit is now the Oracle JDK, instead of the IBM® JDK.

When you upgrade your deployment of i2 Analyze, the IBM JDK is uninstalled and the Oracle JDK is installed.

Information Store processing after you upgrade i2 Analyze

After you upgrade a deployment of some versions of i2 Analyze with an Information Store, extra processing of the data in the Information Store takes place. Analysts can continue to use the system during while the data is processed.

Upgrade processes

There are a number of different processes that can occur after you upgrade i2 Analyze. The processes that are started depend on the version that you are upgrading to or from. Each process places some restrictions on the system before they are completed.

The following processes are completed the first time that you upgrade a deployment with a SQL Server database to version 4.3.4 or later:

- Update multi-line property columns data type from CLOB to VARCHAR
 - Blocks analysts from uploading to the Information Store
 - Blocks ingestion and deletion

The following processes are completed the first time that you upgrade to version 4.3.1 or later:

- Update record identifiers for uploaded records
 - Blocks deletion-by-rule operations
 - Blocks analysts from uploading to the Information Store
- Update record identifiers for ETL records
 - Blocks deletion-by-rule operations
 - Blocks analysts from uploading to the Information Store

The following processes are completed the first time that you upgrade to version 4.2.0 or later:

- Update link end provenance
 - Blocks ingestion and deletion
- Delete link provenance that has missing link ends
 - Blocks ingestion and deletion

Status and progress reports

The `IS_Public.Upgrade_Status` view in the Information Store database shows the list of processes that are complete and pending:

description	status	start_time	end_time
Update record identifiers for uploaded records	Complete	2019-08-13 15:27:06.76	2019-08-13 15:30:03.24

description	status	start_time	end_time
Update record identifiers for ETL records	Pending	2019-08-13 15:30:04.49	

Additionally, the `IS_Public.Upgrade_Progress` view shows the progress of each process by item type:

description	schema_type_id	display_name	status	start_time	end_time
Update record identifiers for uploaded records	ET5	Person	Pending	2019-08-15 22:24:36.72	
Update record identifiers for uploaded records	ET1	Address	Pending		
Update record identifiers for ETL records	ET5	Person	Pending		
Update record identifiers for ETL records	ET1	Address	Pending		

You can use this view to check how many item types are completed and how many are pending. The `IS_Public.Upgrade_Progress` view is populated with the item type status only when the process is in the *Pending* state in the `IS_Public.Upgrade_Status` view.

The possible values for the status columns are:

Pending

The process is either not started or started but not completed.

Complete

The process is completed.

Note: If you stop i2 Analyze before all the processes are complete, the next time that you start i2 Analyze any pending processes continue.

Upgrading a customized Information Store

The Information Store is designed to store large amounts of data, and the underlying database can be customized to optimize performance at scale. If you have modified your database in this manner you must also handle the database upgrade separately.

If you have customized your Information Store, having the deployment toolkit upgrade your database structure automatically is not desirable. However, to upgrade your system, you still need to modify the Information Store to match the newer version.

Note: Depending on the scale and complexity of your data, making changes of this nature can take time. Plan your upgrade to take place in a period of low activity, and back up your system before proceeding.

1. Open a command prompt on the server, and navigate to the `toolkit\scripts` directory of the i2 Analyze 4.3.5 toolkit.

2. To generate the DDL scripts that you can use to upgrade your Information Store, run the following command:

```
setup -t generateInfoStoreUpgradeScripts
```

The upgrade scripts are placed in the following location:

```
toolkit\scripts\database\db2\InfoStore\generated\upgrade
```

3. Evaluate the scripts provided and use them to update your database.

When you have modified your Information Store to match the latest structure, you must ensure that your applications are upgraded without the database upgrade. To do this, ensure that you have set the `create-database` attribute of the Information Store data-source to `false` in the `topology.xml` before [Upgrading an i2 Analyze deployment](#).

Download pdfs

i2 Analyze documentation is also available in PDF documents.

- [Full i2 Analyze documentation](#)
- [Schema Designer](#)

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited Hursley House Hursley Park Winchester, Hants, SO21 2JN UK

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.



Product Number: 5725-G22